Management der Docker-Container für kvwmap

Erstellt von: Peter Korduan, GDI-Service Rostock
letzte Änderung am: 31.05.2017

Änderungen:

Datum	Änderung
09.05.2017	Dokumentation erstellt für kvwmap-server in der Version vom commit: 7900661
10.05.2017	Beschreibung zu PGSQL_DB und Seitennummern korrigiert.
31.05.2017	Restart und Reload hinzugefügt

Inhaltsverzeichnis

1 Einleitung

Ein Container basiert immer auf einem Image. Um einen Container zum laufen zu bringen erzeugt docker zunächst immer erst einen Container und der kann dann gestartet werden. In der Praxis wird das Erzeugen und Starten mit dem Befehl docker run in einem Rutsch durchgeführt. docker run ist aber identisch mit docker create + docker start. Das löschen eines Containers besteht demzufolge aus dem Stoppen und Löschen des Containers, also docker stop + docker rm

1.1 Das Skript dcm

Um dass Management der Container, Erzeugen, Starten, Stoppen, Löschen etc. zu vereinfachen wurde das Skript dcm entwickelt, welches eine Abkürzung für docker container manager ist.

In der Datei dcm ist die Variable OS_DIR angegeben. Dies ist in der Regel /home/gisadmin. Weicht der Pfad ab, muss der in diesem Dokument ersetzt werden. dcm befindet sich in der Datei

```
/home/gisadmin/kvwmap-server/dcm
```

und kann von überall aus auf dem Hostrechner aber nur als root ausgeführt werden. Wird dcm ohne Parameter aufgerufen erscheint eine Liste der möglichen Optionen

Das Skript dcm ist nicht dazu bestimmt angepasst zu werden. Es enthält die Konfiguration und Befehle, die kvwmap benötigt um zu funktionieren. Benutzerspezifische Eigenschaften, wie Passwörter, Pfade auf Dateien und sonstige Konfigurationen für php, apache, postgres etc. werden in gesonderten Konfigurationsdateien verwaltet, siehe 2.

Das Skript dcm enthält nur die Befehle für die Standard-Container, die kvwmap benötigt. Zusätzliche Container für oracle, geoserver, tomcat, pydio, owncloud, etc. sind jedoch in dem Verzeichnis

/home/gisadmin/kvwmap-server/cargo-available

vorkonfiguriert. Durch Aktivierung zusätzlicher Container (cargos), siehe Abschnitt 3.2, stehen auch weitere Befehle zur Verwaltung dieser Cargo-Container in dcm zur Verfügung.

Hinweis:

In diesem Dokument wird der Begriff container-name verwendet. Es handelt sich hierbei um die Bezeichnung des Verzeichnisses in dem die Konfiguration des Containers steckt und die Namen der cargos. Der Name ist nicht zu verwechseln mit dem Namen der Container, die docker verwendet, die Optionsbezeichnungen für das Skript dcm oder die Namen der Container, die beim Linken von fremden Containern in zu startende Container verwendet werden.

In einer folgenden Version werden die Namen der Konfigurationsverzeichnisse, Optionen in dcm und docker Container namen vereinheitlicht.

2 Konfiguration der Standard-Container

Im Unterverzeichnis etc hat jeder Container ein weiteres Unterverzeichnis, welches den Container bezeichnet, z.B. mysql, postgresql, gdal, geoserver oder web.

Die Konfiguration der Container erfolgt in den Dateien env_and_vars, z.B. in

```
/home/gisadmin/etc/[container-name]/env_and_vars
```

Im Folgenden werden die Konfigurationsparameter beschrieben und darunter ein Beispiel für den jeweiligen Container.

2.1 Konfiguration des Container pgsql

Die Konfigurationsdatei findet sich in

```
/home/gisadmin/etc/postgresql/env_and_volumes
```

Konstante	Beschreibung
PGSQL_DB	Initialer Name der Datenbank zum Anlegen des Datenbank-Clusters
PGSQL_USER	Initialer Nutzername zum Anlegen des Datenbank-Clusters
PGSQL_ROOT_PASSWORD	Initiales Passwort zum Anlegen des Datenbank-Clusters
PGSQL_PASSFILE	Name der Passwort-Datei für spätere Nutzung der Datenbank mit kvwmap und gdal
PGSQL_MAJOR_VERSION	Version des postgres-Images
POSTGIS_VERSION	Version des Postgis-Images
PGSQL_IMAGE	Image welches für den Container verwendet werden soll
PGSQL_IMAGE_VERSION	Version des Images welches geladen werden soll

```
#!/bin/bash
PGSQL_DB=postgres
PGSQL_USER=postgres
#read -s -p "Enter Password for PostgreSql user root: " PGSQL_ROOT_PASSWORD
PGSQL_ROOT_PASSWORD=postgres
# PGSQL_PASSFILE must have permission 600
PGSQL_PASSFILE=/root/.pgpass
PGSQL_MAJOR_VERSION=9.4
POSTGIS_VERSION=2.1
PGSQL_IMAGE=mdillon/postgis
PGSQL_IMAGE_VERSION="${PGSQL_MAJOR_VERSION}-${POSTGIS_VERSION}"

pgsql_env_vars="-e \"TERM=xterm\" \
            -e POSTGRES_USER=$PGSQL_USER \
            -e POSTGRES_DB=$PGSQL_DB \
            -e PG_MAJOR=$PGSQL_MAJOR_VERSION \
            -e PGPASSFILE=$PGSQL_PASSFILE"

pgsql_volumes="--volumes-from wwwdata \
           -v $DB_ROOT/postgresql/data:/var/lib/postgresql/data \
           -v $USER_DIR/etc/postgresql/.pgpass:$PGSQL_PASSFILE \
           -v $USER_DIR/etc/proj/epsg:/usr/share/proj/epsg \
           -v $USER_DIR/etc/proj/MVTR2010.gsb:/usr/share/proj/MVTR2010.gsb \
           -v $USER_DIR/etc/proj/MVTRS4283.gsb:/usr/share/proj/MVTRS4283.gsb"
```

2.2 Konfiguration des Container gdal

Die Konfigurationsdatei findet sich in

```
/home/gisadmin/etc/gdal/env_and_volumes
```

Konstante	Beschreibung
GDAL_IMAGE	Image welches für den Container verwendet werden soll
GDAL_IMAGE_VERSION	Version des Images welches geladen werden soll

```
#!/bin/bash
GDAL_IMAGE=pkorduan/gdal-sshd
GDAL_IMAGE_VERSION=2.2.1

# USER_DIR is defined in dcm
# PGSQL_PASSFILE is defined in $USER_DIR/etc/postgresql/volumes loaded previously in dcm
```

```
gdal_env_vars="-e \"TERM=xterm\" \
-e PATH=$PATH:/usr/local/gdal/bin \
-e PGPASSFILE=$PGSQL_PASSFILE"

gdal_volumes="--volumes-from wwwdata \
-v $USER_DIR/etc/postgresql/.pgpass:$PGSQL_PASSFILE"
```

2.3 Konfiguration des Container web

Die Konfigurationsdatei findet sich in

```
/home/gisadmin/etc/web/env_and_volumes
```

Konstante	Beschreibung
KVWMAP_IMAGE	Image welches für den Container verwendet werden soll
KVWMAP_IMAGE_VERSION	Version des Images welches geladen werden soll
KVWMAP_INIT_PASSWORD	Wird nur bei der erstmaligen Einrichtung des Containers verwendet
HTTP_PROXY_WEB	Adresse des Web-Proxy falls vorhanden
NO_PROXY_WEB	Welche Aufrufe nicht über den Proxy laufen sollen
IP_EXTERN	Die IP-Adresse unter der der Server (Host-Rechner) von außen sichtbar ist
DOMAIN_EXTERN	Die Domain unter der der Server (Host-Rechner) von außen sichtbar ist

```
#!/bin/bash
KVWMAP_IMAGE=pkorduan/kvwmap-server
KVWMAP_IMAGE_VERSION="1.1.3"

KVWMAP_INIT_PASSWORD=kvwmap

HTTP_PROXY_WEB=
NO_PROXY_WEB=localhost,pgsql

if [ "$(id -u)" == '0' ]; then
  IP_EXTERN=`ifconfig eth0 | grep 'inet ' | awk '{ print $2 }' | cut -d: -f2`
else
  IP_EXTERN="server_ip"
fi

if [ -f $USER_DIR/etc/apache2/domain ]; then
  DOMAIN_EXTERN=$(head -n 1 $USER_DIR/etc/apache2/domain)
fi

web_env_vars="-e OS_USER=$OS_USER \
            -e IP_EXTERN=$IP_EXTERN \
            -e DOMAIN_EXTERN=$DOMAIN_EXTERN \
            -e KVWMAP_INIT_PASSWORD=$KVWMAP_INIT_PASSWORD \
            -e http_proxy=$HTTP_PROXY_WEB \
            -e no_proxy=localhost,${IP_EXTERN} \
            -e \"TERM=xterm\""
web_volumes="--volumes-from wwwdata \
            -v $USER_DIR/etc/apache2/sites-available:/etc/apache2/sites-available \
            -v $USER_DIR/etc/apache2/sites-enabled:/etc/apache2/sites-enabled \
            -v $USER_DIR/etc/apache2/ssl:/etc/apache2/ssl \
            -v $USER_DIR/etc/php5/apache2/php.ini:/etc/php5/apache2/php.ini \
```

```
    -v $USER_DIR/etc/php5/cli/php.ini:/etc/php5/cli/php.ini \
    -v $USER_DIR/etc/phpmyadmin/config.inc.php:/srv/www/phpmyadmin/config.inc.php \
    -v $USER_DIR/etc/proj/epsg:/usr/share/proj/epsg \
    -v $USER_DIR/etc/proj/MVTR2010.gsb:/usr/share/proj/MVTR2010.gsb \
    -v $USER_DIR/etc/proj/MVTRS4283.gsb:/usr/share/proj/MVTRS4283.gsb"
```

3 Cargo-Container verwalten

Cargo-Container sind zusätzliche Container, die auf dem Host (sozusagen als Fracht) mitlaufen sollen, aber nicht direkt für den Betrieb von kvwmap notwendig sind, z.B. ein Container für geoserver oder eine owncloud für Filesharing. Diese Cargos können, wenn sie selber Web-Services bieten, in den Web-Container gelinkt und somit über den Port 80 oder 443 über Apache mit Redirect angesprochen werden.

Dazu müssen die Cargos zunächst konfiguriert, dann aktiviert und schließlich erzeugt und gestartet werden.

3.1 Cargo-Container konfigurieren

Die Konfiguration eines Cargo-Container erfolgt im Prinzip wie die der Standard-Container. Im Unterverzeichnis von /home/gisadmin/etc mit dem Namen des Containers (container-name) gibt es eine Datei env_and_vars in der das Image, die Version, die Umgebungsvariablen und Volumes definiert sind. Im Folgenden wird die Konfiguration des Cargo-Containers geoserver beschrieben. Beispiele für Konfigurationen von Cargos gibt es auch unter /home/gisadmin/kvwmap-server/[cargo-name-etc]/etc/[cargo-name-etc]. cargo-name-etc entspricht dem container-name des jeweiligen Container.

3.1.1 Konfiguration des Container geoserver

Die Konfigurationsdatei findet sich in

```
/home/gisadmin/etc/geoserver/env_and_volumes
```

Konstante	Beschreibung
GEOSERVER_IMAGE	Image welches für den Container verwendet werden soll
GEOSERVER_IMAGE_VERSION	Version des Images welches geladen werden soll

```
#!/bin/bash

GEOSERVER_IMAGE=pkorduan/geoserver_inspire
GEOSERVER_IMAGE_VERSION=2.9.1

geoserver_volumes="-v $GEOSERVER_DATA_DIR:/opt/geoserver/data_dir"
geoserver_env_vars="-e \"TERM=xterm\""
```

3.2 Verfügbarmachen zusätzlicher Container

Um einen Cargo für das Skript dcm verfügbar zu machen, muss im Verzeichnis

```
/home/gisadmin/kvwmap-server/cargo-enabled
```

ein Link auf das Verzeichnis des cargo-Containers im Verzeichnis

```
 /home/gisadmin/kvwmap-server/cargo-available
```

gesetzt werden, z.B. für geoserver mit:

```
ln -s ../cargo-available/geoserver/
```

Dieser Befehl erzeugt einen Link von

```
/home/gisadmin/kvwmap-server/cargo-enabled/geoserver
```

auf

```
/home/gisadmin/kvwmap-server/cargo-available/geoserver
```

Das Skript dcm durchsucht jedes Mal, wenn es aufgerufen wird das Verzeichnis cargo-enabled nach Dateien mit der Bezeichnung dcm und bindet diese ein.

Ist nun also z.B.

```
/home/gisadmin/kvwmap-server/cargo-enabled/geoserver
```

vorhanden, wird

```
/home/gisadmin/kvwmap-server/cargo-available/geoserver/dcm
```

gefunden und verwendet. Damit sind Befehle wie

```
dcm run geoserver
```

oder

```
dcm rerun geoserver
```

verfügbar und

```
dcm run all
```

erzeugt und startet auch die Cargo-Container.

Hinweis:

Nicht alle Cargo-Container sind getestet und auch für die Cargo-Container gilt eine Reihenfolge für das Starten. Für Rückfragen zur Konfiguration wenden Sie sich bitte an GDI-Service.

4 Erzeugen und Starten der Container

Die Container lassen sich mit dem Befehl

```
dcm run Option
```

starten. Die Optionen werden angezeigt, wenn man den Befehl ohne Option aufruft. Je nach Einstellungen der Cargos sind z.B. folgende Optionen möglich:

```
Gib nach run die Parameter all, gdal, kvwmap, mysql, pgsql, web, wwwdata oder geoserver an.
```

Mit Option „all“ werden alle zur Verfügung stehenden und in cargo-enabled angegebenen Container erzeugt und gestartet.

Mit „kvwmap“ werden alle für kvwmap benötigten Container gestartet. Das sind derzeit die in der folgenden Liste aufgeführten:

- create_www_data_volume
- run_mysql_container
- run_pgsql_container
- run_gdal_container
- run_web_container

Der Befehl „dcm run Option“ erzeugt und startet die angegebenen Container mit „docker run“ und den erforderlichen Parametern, z.B.

```
docker run --name gdal -h geo5-gdal-container --link pgsql-server:pgsql -e "TERM=xterm" -e PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/gisadmin/kvwmap-server:/usr/local/gdal/bin -e PGPASSFILE=/root/.pgpass --volumes-from wwwdata -v /home/gisadmin/etc/postgresql/.pgpass:/root/.pgpass -P --restart=always -d pkorduan/gdal-sshd:2.2.1
```

dcm verkürzt also den Befehl von docker und vereinfacht das Verwalten der Container.

4.1 Alle Container erzeugen und starten

Mit dem Befehl

```
dcm run all
```

lassen sich alle Container in der richtigen Reihenfolge auf ein mal starten.

Beim Starten der Container mit dem Skript dcm werden folgende Schritte ausgeführt:

- Lesen der Variablen wie OS_USER, USER_DIR, WWW_ROOT und DB_ROOT, die im Kopf des Skriptes definiert sind
- Laden der zusätzlichen Container-Konfigurationen (Cargos) load_cargo
- Aufruf der Funktion run_all_container, die alle Container erzeugt und startet

In der Funktion werden die Funktionen zum Starten der einzelnen Container aufgerufen.

- create_www_data_volume Erzeugt den Datenkontainer
- run_mysql_container
- run_oracle_container, falls als Cargo definiert
- run_pgsql_container
- run_gdal_container
- run_geoserver_container, falls als Cargo definiert
- run_pydio_container, falls als Cargo definiert
- run_web_container, in dem Apache und kvwmap läuft

Die jeweiligen Funktionen laden i.d.R. zunächst die Datei env_and_volumes aus dem Verzeichnis.

```
$USER_DIR/etc/[container-name]
```

Die darin gesetzten Variablen werden für den darauffolgenden Befehl zum Anlegen und Starten des Containers verwendet.
Bei einigen Containern wird vorher auch noch die Variable CARGO_WEB_LINKS gesetzt. Dazu wird die Funktion set_cargo_web_links verwendet.

Der Befehl zum erzeugen und starten des Web-Containers sieht folgendermaßen aus:

```
docker run --name web \ -h ${SERVER_NAME}-web-container \
  --link mysql-server:mysql \
  --link pgsql-server:pgsql \
  --link gdal:gdal \
  $CARGO_WEB_LINKS \
  $web_env_vars \
  $web_volumes \
  --add-host=$SERVER_NAME.$DOMAIN_NAME:${IP_EXTERN} \
  --add-host=$SERVER_NAME:${IP_EXTERN} \
  -p 80:80 \
  -p 443:443 \
  --restart=always \
  -d ${KVWMAP_IMAGE}:${KVWMAP_IMAGE_VERSION}
```

Mit –link werden die genannten Container in diesem Container verfügbar gemacht. Verfügbar heißt, dass der angegebene Name (container-name), z.B. pgsql in der Datei /etc/hosts im Container mit der korrekten dazugehörigen IP-Adresse aufgelöst werden kann.
Die CARGO_WEB_LINKS enthalten weitere –link Definitionen der Cargos.
Die Variablen $..._env_vars enthalten jeweils die Umgebungsvariablen, die in dem Container verfügbar sein sollen und $..._volumnes enthalten die Volume-Definitionen des Containers. Beide Variablen wurden in dem Skript env_and_vars des jeweiligen Containers definiert und vorher geladen, siehe oben.
Anschließend erfolgt optional das Hinzufügen weiterer Hostauflösungen und das Portmapping.
--restart=always definiert, dass der Container immer wieder automatisch gestartet werden soll, wenn er nicht läuft. Nach einem Neustart des Hostrechners werden also auch die vorher laufenden Container wieder automatisch gestartet.
Der Parameter -d gibt an, dass der Container im Hintergrund als demon läuft.
Die Variablen ..._IMAGE und ..._IMAGE_VERSION wurden auch in env_and_volumes gesetzt.

4.2 Einen einzelnen Container erzeugen und starten

Um einen einzelnen Container zu starten wird die entsprechende Option angeben, z.B. erzeugt und startet der Befehl

```
dcm run gdal
```

nur den gdal Container.

Es sollte beachtet werden, dass einige Container andere Container voraussetzen. So benötigt z.B. der gdal-Container den pgsql-Container, da gdal in die Datenbank schreiben können soll und somit der Datenbank-Container in den gdal-Container gelinkt werden muss. Der pgsql-Container muss also vor dem gdal-

Container erzeugt und gestartet werden wenn er noch nicht existiert und gestartet werden wenn er existiert, aber gerade nicht läuft.

5 Stoppen der Container

dcm stop [container]

6 Restart von Containern

cm rerun [container]

z.B.

dcm rerun web

7 Reload von Services in Containern

7.1 Web-Container

dcm console web

service apache2 reload

exit

7.2 pgsql-Container

dcm console pgsql

su postgres

/usr/lib/postgresql/$PG_MAJOR/bin/pg_ctl reload

exit

exit