

Grant n. 732590 - H2020-ICT-2016-2017/H2020-ICT-2016-1 

**EW-Shopp Platform**

**Description of ingestion\_lib**

Table of contents

[Chapter 1 Introduction 7](#_Toc27133441)

[1.1 Common 8](#_Toc27133442)

[1.2 Common\_py 8](#_Toc27133443)

[1.3 Building a container 8](#_Toc27133444)

[Chapter 2 Docker containers 9](#_Toc27133445)

[2.1 00\_fetch\_weather\_ecmwf 9](#_Toc27133446)

[2.1.1 Volumes 9](#_Toc27133447)

[2.1.2 Arguments 9](#_Toc27133448)

[2.1.3 Source 10](#_Toc27133449)

[2.1.4 Rancher setup example: 10](#_Toc27133450)

[2.2 01\_uploadtoarango 11](#_Toc27133451)

[2.2.1 Volumes 11](#_Toc27133452)

[2.2.2 Arguments 11](#_Toc27133453)

[2.2.3 Source 11](#_Toc27133454)

[2.2.4 Rancher setup example: 11](#_Toc27133455)

[2.3 04\_extract\_weather\_ecmwf 12](#_Toc27133456)

[2.3.1 Example of parameter settings: 12](#_Toc27133457)

[2.3.2 Volumes 12](#_Toc27133458)

[2.3.3 Arguments 12](#_Toc27133459)

[2.3.4 Source 13](#_Toc27133460)

[2.3.5 Rancher setup example: 13](#_Toc27133461)

[2.4 05\_uploadtoarango\_dual 14](#_Toc27133462)

[2.4.1 Volumes 14](#_Toc27133463)

[2.4.2 Arguments 14](#_Toc27133464)

[2.4.3 Source 14](#_Toc27133465)

[2.4.4 Rancher setup example: 15](#_Toc27133466)

[2.5 06\_fetch\_weather\_owm 16](#_Toc27133467)

[2.5.1 Volumes 16](#_Toc27133468)

[2.5.2 Arguments 16](#_Toc27133469)

[2.5.3 Source 17](#_Toc27133470)

[2.5.4 Rancher setup example: 17](#_Toc27133471)

[2.6 07\_uploadtomysql 18](#_Toc27133472)

[2.6.1 Volumes 18](#_Toc27133473)

[2.6.2 Arguments 18](#_Toc27133474)

[2.6.3 Source 19](#_Toc27133475)

[2.6.4 Rancher setup example: 19](#_Toc27133476)

[2.7 10\_unzip 20](#_Toc27133477)

[2.7.1 Volumes 20](#_Toc27133478)

[2.7.2 Arguments 20](#_Toc27133479)

[2.7.3 Source 20](#_Toc27133480)

[2.7.4 Rancher setup example: 20](#_Toc27133481)

[2.8 11\_tsv2csv 22](#_Toc27133482)

[2.8.1 Volumes 22](#_Toc27133483)

[2.8.2 Arguments 22](#_Toc27133484)

[2.8.3 Source 22](#_Toc27133485)

[2.8.4 Rancher setup example: 22](#_Toc27133486)

[2.9 12\_split 24](#_Toc27133487)

[2.9.1 Volumes 24](#_Toc27133488)

[2.9.2 Arguments 24](#_Toc27133489)

[2.9.3 Source 24](#_Toc27133490)

[2.9.4 Rancher setup example: 24](#_Toc27133491)

[2.10 13\_transform 25](#_Toc27133492)

[2.10.1 Volumes 25](#_Toc27133493)

[2.10.2 Arguments 25](#_Toc27133494)

[2.10.3 Source 25](#_Toc27133495)

[2.10.4 Rancher setup example: 25](#_Toc27133496)

[2.11 14\_toarango 27](#_Toc27133497)

[2.11.1 Volumes 27](#_Toc27133498)

[2.11.2 Arguments 27](#_Toc27133499)

[2.11.3 Source 27](#_Toc27133500)

[2.11.4 Rancher setup example: 27](#_Toc27133501)

[2.12 15\_jot\_categorize 28](#_Toc27133502)

[2.12.1 Volumes 28](#_Toc27133503)

[2.12.2 Arguments 28](#_Toc27133504)

[2.12.3 Source 29](#_Toc27133505)

[2.12.4 Rancher setup example: 29](#_Toc27133506)

[2.13 16\_jot\_aggregate\_kwds 30](#_Toc27133507)

[2.13.1 Volumes 30](#_Toc27133508)

[2.13.2 Arguments 30](#_Toc27133509)

[2.13.3 Source 30](#_Toc27133510)

[2.13.4 Rancher setup example: 30](#_Toc27133511)

[2.14 17\_jot\_closest\_kwds 31](#_Toc27133512)

[2.14.1 Volumes 31](#_Toc27133513)

[2.14.2 Arguments 31](#_Toc27133514)

[2.14.3 Source 32](#_Toc27133515)

[2.14.4 Rancher setup example: 32](#_Toc27133516)

# Introduction

The components in this library are made as a set of Docker containers that are configured as a processing pipeline. The container processes are passing data through shared directories passed as volumes into the process. The directories are managed by the host OS. The host OS can provide file sharing using distributed file system. If distribution is supported, the container processes can be distributed at different processing units.

The container processes are orchestrated using Rancher for easy management. The Docker containers don't have direct usage of Rancher services and should be possible to start using any orchestration system.

When starting a container process, it is given three volumes in, work and out.

A container process normally polls for input files at its in volume. When a new input file is detected it is moved to the work volume, processed and the result is passed to the out volume.

Container processes will pass data to each other when out volume for one process is mapped to the same directory as in volume for another process.

It is important that the OS user for the processes have access to the volumes they are given.

Currently the consists of the following Docker containers:

* 00\_fetch\_weather\_ecmwf
* 01\_uploadtoarango
* 04\_extract\_weather\_ecmwf
* 05\_uploadtoarango\_dual
* 06\_fetch\_weather\_owm
* 07\_uploadtomysql
* 10\_unzip
* 11\_tsv2csv
* 12\_split
* 13\_transform
* 14\_toarango
* 15\_jot\_categorize
* 16\_jot\_aggregate\_kwds
* 17\_jot\_closest\_kwds

## Common

The implementation of the different containers are based on a common set of scripts that are included by all. Docker does not support files used by several containers. The only way to handle this is to copy the scripts before building. This is done by the 'make\_docker.sh' script.

'main.sh' is the main entry point for the container. It sets up a worker process 'main\_worker.sh' running in the background. The main supports graceful shutdown by capturing SIGTERM and SIGINT. It signals to the worker process to stop by deleting the run-file. The main waits until worker process terminates. This arrangement allows the worker to finalize processing of a file before terminating.  
If the container specific script 'single\_shot.sh' is present this file will be called and the main loop will not be started.

'main\_worker.sh' is the main loop polling for an input file each 60 seconds. If no file is found a wait\_count is incremented. When it reached retry\_max\_count the main loop will terminate. Retry\_max\_count can be set to 0 to disable this functionality. The actual polling is done by 'fetch\_and\_process.sh'.

'fetch\_and\_process.sh' checks for input file based on the file pattern given by the container specific script 'get\_input\_file\_spec.sh'. If file(s) are found one of them is chosen and moved to work. If the container specific script 'check\_input\_file.sh' is present the, file is tested before its moved. During the whole operation the in folder is locked using a lock file. The lock file avoids race conditions caused by other containers accessing the same file. After the file is moved to work the container specific script 'process\_job.sh' is called for processing and finally provide an output file.

'process\_job.sh' can call the script 'move\_to\_output.sh' to safely move one or more files to the out folder using a lock file. The lock file avoids race conditions caused by other containers accessing the same file.

## Common\_py

Common python files can be in this folder. Docker does not support files used by several containers. The only way to handle this is to copy the scripts before building. This is done by the 'make\_docker.sh' script.

## Building a container

Change directory to the 'Dockerfile' for the container to build.  
Call the build script for copying common files, building and pushing the container.

Example:

../common/make\_docker.sh "datagraft/lib\_17\_jot\_closest\_kwds"

# Docker containers

## 00\_fetch\_weather\_ecmwf

This script fetch GRIB weather data for Germany from ECMWF service. Based on the GRIB file it extracts weather data for a set of regions listed in a csv. The extracted data are moved to the out directory.

The time range to fetch data for is given in a job-description-file in json format. This makes it possible to have this container waiting for jobs and fetch weather data on a periodic basis.

Example job file:

{

"start\_date": "2018-05-31",

"end\_date": "2018-05-31"

}

TODO: Make country and region-csv file environment variables.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)
* /root/.ecmwfapirc .. ecmwf credentials must be present in this file

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)
* Full path to the regions csv-file

Example params: 0 /code /in /work /out /coordinates/bbox-csv

To enable logging set the environment variable LOG\_JOBS to 1

### Source

Docker image: datagraft/lib\_00\_fetch\_weather\_ecmwf

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/00_fetch_weather_ecmwf>

### Rancher setup example:

TBC

## 01\_uploadtoarango

This script uploads a json file to an Arango database using HTTP API. Credentials are given in arguments.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)
* Server\_url
* Username and password
* Db\_name
* Collection\_name

Example params: 0 /code /in /work /out 192.168.4.17:1234 root:xxx weather germany

To enable logging set the environment variable LOG\_JOBS to 1

### Source

Docker image: datagraft/lib\_01\_uploadtoarango

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/01_uploadtoarango>

### Rancher setup example:

TBD

## 04\_extract\_weather\_ecmwf

This script fetches the region weather forecast stored in the grib files and transforms that in a CSV (default) or json forecast for the cities inputed in an input CSV file.

### Example of parameter settings:

WE\_INPUT\_TYPE=env

WE\_START\_DATE=2018-05-14

WE\_END\_DATE=2018-05-14

WE\_GRIB\_FILE=path-to/weather-slovenia\_m5\_y2018.grib

WE\_FORECAST\_DAYS=16

WE\_KEY\_SEQUENCE=2d,2t,10u,10v,ptype,sd,sf,sund,ssr,sp,tcc,tp,vis,ws,rh,cityName,region,strRegion,geonameId,position,validTime,validityDateTime

WE\_NEW\_KEY\_SEQUENCE=,,,,,,,,,,,,,,,,,,,,,

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)

Example params: 0 /code /in /work /out

To enable logging set the environment variable LOG\_JOBS to 1

The takes as arguments:

* the coordinates of the cities to which one wants the forecast
* the name of the outputfile (without extension)
* (optionally) the name of the input json file with the forecast parameters

The forecast parameters can be passed via a json file or via environment variables. The default is via environment variables. If you want the parameters to be passed via json, you must set the environment variable WE\_INPUT\_TYPE to "json". If the variable is not set or set to "env", the script will take the parameters from environment variables.

Below is a description of the parameters to be passed via environment variables:

* WE\_INPUT\_TYPE: to be set to "env" or "json". "env" means forecast parameters inputed via environment variables and "json" via json file.
* WE\_START\_DATE: start date in format YYYY-MM-DD
* WE\_END\_DATE: end date in format YYYY-MM-DD
* WE\_GRIB\_FILE: grib file with the forecast
* WE\_FORECAST\_DAYS: number of forecast days
* WE\_KEY\_SEQUENCE: specify which weather forecast fields of the grib will be shown on the csv and their order. The fields must be separated by commas (no spaces)
* WE\_NEW\_KEY\_SEQUENCE: this is used to possibly rename the forecast fields when writting the csv. It must constain the same number of elements as the WE\_KEY\_SEQUENCE. And be empty for the fields not to be renamed. The position field will be expanded and renamed to "Latitude" and "Longitude" regardless of the WE\_NEW\_KEY\_SEQUENCE.
* WE\_JSON\_OUTPUT: if set to true, the output forecast will be written to a json file instead of a cvs. The default is false.

### Source

Docker image: datagraft/lib\_04\_extract\_weather\_ecmwf

Source code: https://github.com/ew-shopp/ingestion\_lib/tree/master/04\_extract\_weather\_ecmwf

### Rancher setup example:

TBC

## 05\_uploadtoarango\_dual

This script uploads a json file to an Arango database using HTTP API. It detects whether the file is destinated for a document collection or an edge collection and uploads to the correct collection. Credentials are given in arguments.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)
* Server\_url
* Username and password
* Db\_name
* Doc\_collection\_name
* Edge\_collection\_name
* From\_prefix\_name
* To\_prefix\_name

Example params: 0 /code /in /work /out 192.168.4.17:1234 root:xxx weather Germany Germany\_edge xx yy

To enable logging set the environment variable LOG\_JOBS to 1

### Source

Docker image: datagraft/lib\_05\_uploadtoarango\_dual

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/05_uploadtoarango_dual>

### Rancher setup example:

TBD

## 06\_fetch\_weather\_owm

This script fetches weather data for city locations specified in CSV file. Each city is a separate request to OWM (OpenWeatherMap) and results in todays forcast is stored in a JSON file. The forecast for all cities is collected and transformed that in one CSV (default) or json file. The process does run as a one-shot, fetching weather data and terminates. It will be started by a CRON job at Monday once a week '0 1 \* \* 1'. The process input location is not polled. More information about CRON setup can be found at [https://crontab.guru/#0\_1\_\*\_\*\_1](https://crontab.guru/#0_1_*_*_1)

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)
* /coordinates..- location for location CSV file (match path in environment)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)

Example params: 0 /code /in /work /out

The forecast parameters are passed via environment variables:

* WE\_INPUT\_TYPE: to be set to "env"
* WE\_OWM\_KEY: OWM API key used when accessing the weather service
* WE\_REGION\_CSV\_FILE: csv file listing all the regions and their coordinates
* WE\_KEY\_SEQUENCE: specify which weather forecast fields of the grib will be shown on the csv and their order. The fields must be separated by commas (no spaces)
* WE\_NEW\_KEY\_SEQUENCE: this is used to possibly rename the forecast fields when writing the csv. It must contain the same number of elements as the WE\_KEY\_SEQUENCE. And be empty for the fields not to be renamed. The position field will be expanded and renamed to "Latitude" and "Longitude" regardless of the WE\_NEW\_KEY\_SEQUENCE.
* WE\_JSON\_OUTPUT: if set to true, the output forecast will be written to a json file instead of a csv. The default is false.

Example of environment vars:

WE\_INPUT\_TYPE: env

WE\_OWM\_KEY: lkljrjwerlkj134u9024

WE\_REGION\_CSV\_FILE: /coordinates/regions\_measurence.csv

WE\_KEY\_SEQUENCE: 2d,2t,10u,10v,ptype,sd,sf,sund,ssr,sp,tcc,tp,vis,ws,rh,cityName,region,strRegion,geonameId,position,validTime,validityDateTime

WE\_NEW\_KEY\_SEQUENCE: ',,,,,,,,,,,,,,,,,,,,,'

### Source

Docker image: datagraft/lib\_06\_fetch\_weather\_owm

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/06_fetch_weather_owm>

### Rancher setup example:

test-owm:

image: datagraft/test\_06\_fetch\_weather\_owm

environment:

WE\_INPUT\_TYPE: env

WE\_OWM\_KEY: lkljrjwerlkj134u9024

WE\_REGION\_CSV\_FILE: /coordinates/regions\_measurence.csv

WE\_KEY\_SEQUENCE: 2d,2t,10u,10v,ptype,sd,sf,sund,ssr,sp,tcc,tp,vis,ws,rh,cityName,region,strRegion,geonameId,position,validTime,validityDateTime

WE\_NEW\_KEY\_SEQUENCE: ',,,,,,,,,,,,,,,,,,,,,'

stdin\_open: true

volumes:

- /bigdata/steffen//run\_daily\_weather\_owm/in:/in

- /bigdata/steffen//run\_daily\_weather\_owm/\_work00:/work

- /bigdata/steffen//run\_daily\_weather\_owm/\_out00:/out

- /bigdata/steffen//run\_daily\_weather\_owm/coordinates:/coordinates

tty: true

user: 1000:1000

command:

- '0'

- /code

- /in

- /work

- /out

labels:

io.rancher.container.start\_once: 'true'

io.rancher.container.pull\_image: always

cron.schedule: 0 1 \* \* 1

## 07\_uploadtomysql

This script fetches a csv file with one header row and upload the data to a MySQL database table  
The script monitors the input dir for csv files to upload. If not file present it will sleep for 60sec before trying again.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - used for passing input files to this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (how many idle 60 sleep periods before terminating)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)

Example params: 0 /code /in /work /out

A set of environment variables are needed to access the database

* WE\_SERVER\_ADDR: IP address of name to the MySQL server
* WE\_SERVER\_USR: Username
* WE\_SERVER\_PWD: Password
* WE\_DB\_NAME: Database name.
* WE\_TABLE\_NAME: Table name in the database to upload to.
* WE\_TABLE\_COLUMNS: Column names to use. Has to be a comma separated list.

Example of environment vars:

WE\_SERVER\_ADDR=192.168.1.21

WE\_SERVER\_USR=my\_user

WE\_SERVER\_PWD=xyzz

WE\_DB\_NAME=test\_import

WE\_TABLE\_NAME=weather\_data

WE\_TABLE\_COLUMNS=(storeId, date, temp\_avg, temp\_max, temp\_min, precip\_tot, cloud\_cov\_avg)

### Source

Docker image: datagraft/lib\_07\_uploadtomysql

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/07_uploadtomysql>

### Rancher setup example:

test-mysql-upload:

image: datagraft/test\_07\_uploadtomysql

environment:

WE\_SERVER\_ADDR: 192.168.1.21

WE\_SERVER\_USR: root

WE\_SERVER\_PWD: xyzz

WE\_DB\_NAME: test\_import

WE\_TABLE\_NAME: weather\_data

WE\_TABLE\_COLUMNS: (storeId, date, temp\_avg, temp\_max, temp\_min, precip\_tot, cloud\_cov\_avg)

LOG\_JOBS: '1'

stdin\_open: true

volumes:

- /bigdata/steffen//run\_daily\_weather\_owm/\_out01:/in

- /bigdata/steffen//run\_daily\_weather\_owm/\_work02:/work

- /bigdata/steffen//run\_daily\_weather\_owm/\_out02:/out

tty: true

user: 1000:1000

command:

- '0'

- /code

- /in

- /work

- /out

labels:

io.rancher.container.pull\_image: always

io.rancher.container.start\_once: 'true'

## 10\_unzip

This script unzip a zip-archive without folder information. The files in the archive are moved to the out directory.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)

Example params: 0 /code /in /work /out

To enable logging set the environment variable LOG\_JOBS to 1

### Source

Docker image: datagraft/test\_06\_fetch\_weather\_owm

Source code: <https://github.com/ew-shopp/weather_import/tree/master/docker_daily_import/06_fetch_weather_owm>

### Rancher setup example:

00-Unzip:

image: datagraft/lib\_10\_unzip

environment:

LOG\_JOBS: '1'

stdin\_open: true

volumes:

- /bigdata/steffen/run\_bc4\_cat\_germany/in:/in

- /bigdata/steffen/run\_bc4\_cat\_germany/\_work00:/work

- /bigdata/steffen/run\_bc4\_cat\_germany/\_out00:/out

tty: true

user: 1000:1000

command:

- '0'

- /code

- /in

- /work

- /out

labels:

io.rancher.container.pull\_image: always

io.rancher.container.start\_once: 'true'

## 11\_tsv2csv

This script converts a tab-separated datafile to a comma-separated datafile. The converted files are moved to the out directory.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)

Example params: 0 /code /in /work /out

To enable logging set the environment variable LOG\_JOBS to 1

### Source

Docker image: datagraft/lib\_11\_tsv2csv

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/11_tsv2csv>

### Rancher setup example:

01-tsv2csv:

image: datagraft/lib\_11\_tsv2csv

environment:

LOG\_JOBS: '1'

stdin\_open: true

volumes:

- /bigdata/steffen/run\_bc4\_cat\_germany/\_out00:/in

- /bigdata/steffen/run\_bc4\_cat\_germany/\_work01:/work

- /bigdata/steffen/run\_bc4\_cat\_germany/in\_all\_kwds:/out

tty: true

user: 1000:1000

command:

- '0'

- /code

- /in

- /work

- /out

labels:

io.rancher.container.pull\_image: always

io.rancher.container.start\_once: 'true'

## 12\_split

This script splits a csv file if its above 800000 rows. The header row is not duplicated. The splitted files are moved to the out directory.

TODO: Make number of rows an environment variable

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)

Example params: 0 /code /in /work /out

To enable logging set the environment variable LOG\_JOBS to 1

### Source

Docker image: datagraft/lib\_11\_tsv2csv

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/12_split>

### Rancher setup example:

TBC

## 13\_transform

This script fetches a csv file and applies a transformation to it. The transformation is given as a .jar file generated by the Grafterizer system.   
The script monitors the input dir for csv files to upload. If not file present it will sleep for 60sec before trying again.  
Depending on the size of the transformation input the container may require memory reservation at startup.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - used for passing input files to this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)
* /transformation..- location for transformation JAR file (match path in argument)

### Arguments

Arguments are passed to the container entrypoint:

Container entrypoint params:

* Max wait for file count. Set to '0' (how many idle 60 sleep periods before terminating)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)
* Path to transformation. Set to '/transformation/xxx.jar' (must match given volume name)

Example params: 0 /code /in /work /out /transformation/transformation.jar

### Source

Docker image: datagraft/lib\_13\_transform

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/13_transform>

### Rancher setup example:

transform-owm-data:

image: datagraft/grafterizer-jar-executor

stdin\_open: true

volumes:

- /bigdata/steffen/run\_daily\_weather\_owm/\_out00:/in

- /bigdata/steffen/run\_daily\_weather\_owm/\_work01:/work

- /bigdata/steffen/run\_daily\_weather\_owm/\_out01:/out

- /bigdata/steffen/run\_daily\_weather\_owm/transformation:/transformation

tty: true

mem\_reservation: 4194304000

user: 1000:1000

command:

- '0'

- /code

- /in

- /work

- /out

- /transformation/transformation.jar

labels:

io.rancher.container.pull\_image: always

## 14\_toarango

Transforms a rdf file to doc and edge collections based on an external JS-script. See <https://github.com/datagraft/datagraft-csv-to-arangodb> for details.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)
* /transformation..- location for transformation scripts (match path in argument)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)
* Path to transformation. Set to '/transformation/xxx.js' (must match given volume name)

Example params: 0 /code /in /work /out /transformation/transformscript.js

To enable logging set the environment variable LOG\_JOBS to 1

### Source

Docker image: datagraft/lib\_14\_toarango

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/14_toarango>

### Rancher setup example:

TBC

## 15\_jot\_categorize

This script correlates keywords in the input file to categories given as startup information. The categoriser is documented in <https://github.com/JozefStefanInstitute/ew-shopp-public/tree/master/keyword_clustering>

The keyword to category mapping are normalized and joined with the original input file. The result is a new file with category columns added.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)

Example params: 0 /code /in /work /out

To enable logging set the environment variable LOG\_JOBS to 1

The following parameters are passed via environment variables:

* JOB\_CAT\_FASTEXT\_FULL\_PATH: Path to the FastText model binary file.
* JOB\_CAT\_EMBEDDER\_FULL\_PATH: Path to the embedder parameters json file.
* JOB\_CAT\_CATEGORIES\_FULL\_PATH: Path to the categories file.
* JOB\_CAT\_CATEGORIES\_COLUMN: Name of column containing categories in the categories csv file. (default: Category)
* JOB\_CAT\_CATEGORIES\_ID\_COLUMN: Name of column containing category ids in the categories csv file. (default: CategoryID).
* JOB\_CAT\_KEYWORDS\_COUNTRY\_CODE: Country code to be used from the keywords file.
* JOB\_CAT\_KEYWORDS\_CHUNK\_SIZE: (Optional param) Number of keywords to categorize in one operation. Default 100000, reduce if memory issues.
* JOB\_NOR\_JAR\_FULL\_PATH: Full path to normalized jar file from grafterizer

### Source

Docker image: datagraft/lib\_15\_jot\_categorize

Source code: <https://github.com/ew-shopp/ingestion_lib/tree/master/15_jot_categorize>

### Rancher setup example:

02-categoriser:

image: datagraft/lib\_15\_jot\_categorize

environment:

LOG\_JOBS: '1'

JOB\_CAT\_FASTEXT\_FULL\_PATH: /categoriser/cc.de.300.bin

JOB\_CAT\_EMBEDDER\_FULL\_PATH: /categoriser/de\_embedder.json

JOB\_CAT\_CATEGORIES\_FULL\_PATH: /categoriser/productsservices.csv

JOB\_CAT\_CATEGORIES\_COLUMN: Category

JOB\_CAT\_CATEGORIES\_ID\_COLUMN: Criterion ID

JOB\_CAT\_KEYWORDS\_COUNTRY\_CODE: DE

JOB\_NOR\_JAR\_FULL\_PATH: /categoriser/normalize-clusters.jar

stdin\_open: true

volumes:

- /bigdata/steffen/run\_bc4\_cat\_germany/\_out01:/in

- /bigdata/steffen/run\_bc4\_cat\_germany/\_work02:/work

- /bigdata/steffen/run\_bc4\_cat\_germany/\_out02:/out

- /bigdata/steffen/run\_bc4\_cat\_germany/categoriser:/categoriser

tty: true

mem\_reservation: 15728640000

user: 1000:1000

command:

- '0'

- /code

- /in

- /work

- /out

labels:

io.rancher.container.pull\_image: always

io.rancher.container.start\_once: 'true'

## 16\_jot\_aggregate\_kwds

This script aggregate keywords from all inputfiles into a common list of unique keywords. The output file is used for finding the most popular keywords for a set of categories

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)

Example params: 0 /code /in /work /out

To enable logging set the environment variable LOG\_JOBS to 1

The following parameters are passed via environment variables:

* JOB\_CAT\_KEYWORDS\_COUNTRY\_CODE: Country code to be used from the keywords file.

### Source

Docker image: datagraft/lib\_16\_jot\_aggregate\_kwds

Source code: https://github.com/ew-shopp/ingestion\_lib/tree/master/16\_jot\_aggregate\_kwds

### Rancher setup example:

10-all-kwds:

image: datagraft/lib\_16\_jot\_aggregate\_kwds

environment:

LOG\_JOBS: '1'

JOB\_CAT\_KEYWORDS\_COUNTRY\_CODE: DE

stdin\_open: true

volumes:

- /bigdata/steffen/run\_bc4\_cat\_germany/in\_all\_kwds:/in

- /bigdata/steffen/run\_bc4\_cat\_germany/work\_all\_kwds:/work

- /bigdata/steffen/run\_bc4\_cat\_germany/out\_all\_kwds:/out

tty: true

user: 1000:1000

command:

- '10'

- /code

- /in

- /work

- /out

labels:

io.rancher.container.pull\_image: always

io.rancher.container.start\_once: 'true'

## 17\_jot\_closest\_kwds

This script finds the n closest keywords in the input file for all categories given as startup information. The categoriser is documented in <https://github.com/JozefStefanInstitute/ew-shopp-public/tree/master/keyword_clustering>

The keyword to category mapping are normalized.

### Volumes

Volume setup. Must match arguments:

* /code - internal location for scripts in the container (created internally)
* /in - not used by this container (mandatory)
* /work - used for intermediate files (mandatory)
* /out..-..location for process results for next step in the pipeline (mandatory)

### Arguments

Arguments are passed to the container entrypoint and as environment variables:

Container entrypoint params:

* Max wait for file count. Set to '0' (not used by this container)
* Name of code location. Set to '/code'
* Name of input location. Set to '/in' (must match given volume name)
* Name of work location. Set to '/work' (must match given volume name)
* Name of out location. Set to '/out' (must match given volume name)

Example params: 0 /code /in /work /out

To enable logging set the environment variable LOG\_JOBS to 1

The following parameters are passed via environment variables:

* JOB\_CAT\_ALL\_KEYWORDS\_FULL\_PATH: Path to file with all keywords.
* JOB\_CAT\_FASTEXT\_FULL\_PATH: Path to the FastText model binary file.
* JOB\_CAT\_EMBEDDER\_FULL\_PATH: Path to the embedder parameters json file.
* JOB\_CAT\_CATEGORIES\_FULL\_PATH: Path to the categories file.
* JOB\_CAT\_CATEGORIES\_COLUMN: Name of column containing categories in the categories csv file. (default: Category)
* JOB\_CAT\_CATEGORIES\_ID\_COLUMN: Name of column containing category ids in the categories csv file. (default: CategoryID).
* JOB\_CAT\_N\_KEYWORDS: Number of closest keywords to match each category.

### Source

Docker image: datagraft/lib\_17\_jot\_closest\_kwds

Source code: https://github.com/ew-shopp/ingestion\_lib/tree/master/17\_jot\_closest\_kwds

### Rancher setup example:

11-closest-kwds:

image: datagraft/lib\_17\_jot\_closest\_kwds

environment:

LOG\_JOBS: '1'

JOB\_CAT\_FASTEXT\_FULL\_PATH: /categoriser/cc.de.300.bin

JOB\_CAT\_EMBEDDER\_FULL\_PATH: /categoriser/de\_embedder.json

JOB\_CAT\_CATEGORIES\_FULL\_PATH: /categoriser/productsservices.csv

JOB\_CAT\_CATEGORIES\_COLUMN: Category

JOB\_CAT\_CATEGORIES\_ID\_COLUMN: Criterion ID

JOB\_CAT\_N\_KEYWORDS: '1000'

JOB\_CAT\_ALL\_KEYWORDS\_FULL\_PATH: /in/unique\_keywords.csv

stdin\_open: true

volumes:

- /bigdata/steffen/run\_bc4\_cat\_germany/out\_all\_kwds:/in

- /bigdata/steffen/run\_bc4\_cat\_germany/work\_closest\_kwds:/work

- /bigdata/steffen/run\_bc4\_cat\_germany/out\_closest\_kwds:/out

- /bigdata/steffen/run\_bc4\_cat\_germany/categoriser:/categoriser

tty: true

mem\_reservation: 15728640000

user: 1000:1000

command:

- '0'

- /code

- /in

- /work

- /out

labels:

io.rancher.container.pull\_image: always

io.rancher.container.start\_once: 'true'

02-categoriser:

image: datagraft/lib\_15\_jot\_categorize

environment:

LOG\_JOBS: '1'

JOB\_CAT\_FASTEXT\_FULL\_PATH: /categoriser/cc.de.300.bin

JOB\_CAT\_EMBEDDER\_FULL\_PATH: /categoriser/de\_embedder.json

JOB\_CAT\_CATEGORIES\_FULL\_PATH: /categoriser/productsservices.csv

JOB\_CAT\_CATEGORIES\_COLUMN: Category

JOB\_CAT\_CATEGORIES\_ID\_COLUMN: Criterion ID

JOB\_CAT\_KEYWORDS\_COUNTRY\_CODE: DE

JOB\_NOR\_JAR\_FULL\_PATH: /categoriser/normalize-clusters.jar

stdin\_open: true

volumes:

- /bigdata/steffen/run\_bc4\_cat\_germany/\_out01:/in

- /bigdata/steffen/run\_bc4\_cat\_germany/\_work02:/work

- /bigdata/steffen/run\_bc4\_cat\_germany/\_out02:/out

- /bigdata/steffen/run\_bc4\_cat\_germany/categoriser:/categoriser

tty: true

mem\_reservation: 15728640000

user: 1000:1000

command:

- '0'

- /code

- /in

- /work

- /out

labels:

io.rancher.container.pull\_image: always

io.rancher.container.start\_once: 'true'