# mosaik-docker

*Release 0.1*

**Dec 01, 2020**

# Usage

The *mosaik-docker* package eases the deployment of the mosaik co-simulation framework with Docker. This package provides:

- an extension to the mosaik framework that allows to start and run simulators as separate Docker containers
- a command line interface (CLI) that eases the deployment of dockerized mosaik simulations

Examples of *mosaik-docker simulation setups* can be found here.

The *mosaik-docker* package can also be used with JupyterLab, please visit the mosaik-docker JupyterLab extension for more information on this subject.

CHAPTER 1

# Usage

Find information about how to use the *mosaik-docker* package *here*.

## 1.1 Using package mosaik-docker

### 1.1.1 Overview

Package *mosaik-docker* eases the deployment of the mosaik co-simulation framework with Docker. A typical *mosaik-docker* workflow contains the following steps:

1. **Create a simulation setup**: A *simulation setup* is a directory that contains all necessary scripts and configuration files. Package *mosaik-docker* provides a command to create a "bare" simulation setup that needs be adapted to your application. To do so, you need to provide the following:

- Provide the *mosaik scenario file* for your co-simulation.
- Provide the *Dockerfile(s)* for the *mosaik sim manager* (and optionally also other simulators).

2. **Configure simulation setup**: Specify the mosaik scenario file and Dockerfile(s) to be used for your simulation setup. You can also add additional input files or folder and specify output files.
3. **Check and build simulation setup**: Check if your simulation setup is valid and build the Docker images for running the simulations.
4. **Run simulations and check their status**: You can start / stop new simulation runs and check their current execution status.
5. **Retrieve simulation results**: After a simulations has successfully finished, you can retrieve its results.

In the following, these steps are explained in more detail. For each step, the required commands (for the command line terminal) are explained. If you want to use *mosaik-docker* via its graphical user interface, you can find more information here.

### 1.1.2 Create a simulation setup

A *simulation setup* is a directory that contains all necessary scripts and configuration files. Package *mosaik-docker* provides the `create_sim_setup` command to create a template for a simulation setup that can be adapted to your needs (see here for details):

```
>>> create_sim_setup MySimSetup
Created new simulation setup: /home/user/MySimSetup
```

This will create a new directory to which you have to add the following:

- a mosaik scenario that
  1. starts the simulators
  2. instantiates models within the simulators
  3. connects the model instances of different simulators to establish the data flow between them
- a Dockerfile for running sim manager (i.e., the mosaik orchestrator that executes the scenario)
- Dockerfiles for running the simulators (optional)
- input files and/or folders (optional)

Examples of a *monolithic simulation setup* (the mosaik sim manager and all simulators run in the same Docker container) and a *distributed simulation setup* (the mosaik sim manager and the simulators run in individual Docker containers) can be found here.

### 1.1.3 Configure simulation setup

The configuration for the simulation setup is stored in file `mosaik-docker.json`. All required information to run a dockerized mosaik simulation is stored in this file (mosaik scenario file, Dockerfile(s), input files and/or folder, output files). Package *mosaik-docker* provides the `configure_sim_setup` command for the configuration of simulation setups (see here for details):

```
>>> configure_sim_setup --scenario-file main.py --docker-file dockerfiles/Dockerfile_
↪main --extra-file demo_lv_grid.json --result demo.hdf5
Updated simulation configuration file: /home/user/MySimSetup/mosaik-docker.json
```

**NOTE**: It is highly recommended to NOT edit this configuration file by hand, but use the commands provided my *mosaik-docker*!

### 1.1.4 Check and build simulation setup

You can use command `check_sim_setup` check if your simulation setup is valid (see here for details):

```
>>> check_sim_setup
simulation setup is valid: /home/user/MySimSetup
```

Once your setup seems to be fine, you can use command `build_sim_setup` to build the Docker images for running your simulation (see here for details):

```
>>> build_sim_setup
Sending build context to Docker daemon    701kB
Step 1/11 : FROM mosaik/orch-base:v1
 ---> b239b8430a38
```

(continues on next page)

(continued from previous page)

```
Step 2/11 : ARG SCENARIO_FILE
 ---> Using cache
 ---> 8df59427a94c
Step 3/11 : ARG EXTRA
 ---> Using cache
 ---> 57355196ae2a
Step 4/11 : RUN pip install mosaik-csv==1.0.3
 ---> Using cache
 ---> 5f98b074eccc
Step 5/11 : RUN pip install mosaik-hdf5==0.3
 ---> Using cache
 ---> 96daeb62dee8
Step 6/11 : RUN pip install mosaik-householdsim==2.0.3
 ---> Using cache
 ---> 821122dc1287
Step 7/11 : RUN pip install mosaik-pypower==0.7.2
 ---> Using cache
 ---> 06dea0bd92ca
Step 8/11 : RUN pip install networkx==2.4
 ---> Using cache
 ---> c8fdf48dfd2e
Step 9/11 : COPY $SCENARIO_FILE .
 ---> Using cache
 ---> 15f73891d199
Step 10/11 : COPY $EXTRA .
 ---> 2e3050d9fe48
Step 11/11 : ENTRYPOINT python $SCENARIO_FILE
 ---> Running in a428eddbdfba
Removing intermediate container a428eddbdfba
 ---> 90db1e9de30b
Successfully built 90db1e9de30b
Successfully tagged mosaik/orch/my-sim-setup:latest
building simulation setup succeeded: /home/user/MySimSetup
```

### 1.1.5 Run simulations and check their status

Once the Docker images have been successfully built, you can use command `start_sim` to start new simulation runs (see here for details):

```
>>> start_sim
ddf4398dacaa0208ac6a0b4c7c4c482a981aa7fa7d458fca578eb31728f0e735
Started new simulation with ID = d150f4
```

Use command `get_sim_status` to check the current execution status of your simulations (see here for details):

```
>>> get_sim_status
running:
        d150f4: Up 1 second
finished:
        766540: Exited (0) 18 seconds ago
```

### 1.1.6 Retrieve simulation results

After a simulation has successfully finished, you can use command `get_sim_results` to retrieve the corresponding results (see here for details):

```
>>> get_sim_results --id 766540
Retrieved results for simulation(s) with ID = 766540
```

CHAPTER 2

# Installation

Find information about the requirements and the installation of the *mosaik-docker* package *here*.

## 2.1 Prerequisites

You will need Python (tested with version >= 3.6) to install the extension. For the extension to work properly, you will also need a working installation of Docker Engine.

## 2.2 Installation

The package is available via the official Python Package Index. Install it from the command line:

```
pip install mosaik-docker
```

## 2.3 Troubleshoot

In case you get error messages similar to the following one:

Check if the user has been added to group `docker`.

CHAPTER 3

# Command line interface reference

Find detailed information about the *mosaik-docker* package commands *here*.

## 3.1 Command line interface reference

Package *mosaik-docker* provides the following commands via the console.

### 3.1.1 Commands for handling simulation setups

#### create_sim_setup

Create an empty simulation setup in a new directory:

```
create_sim_setup [-h] [--id ID] NAME [DIR]
```

**Positional arguments**:

- `NAME`: name of the new simulation setup
- `DIR`: directory to put the generated simulation setup (default: current working directory)

**Optional arguments**:

- `-h`, `--help`: show the help message and exit
- `--id ID`: unique ID for the new simulation setup

#### configure_sim_setup

Configure an existing simulation setup:

```
configure_sim_setup [-h] -d DOCKER_FILE -s SCENARIO_FILE \
   [--extra-file EXTRA_FILE] [--extra-dir EXTRA_DIR] \
   [--result RESULT] [SETUP_DIR]
```

**Positional arguments**:

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)

**Required named arguments**:

- `-d DOCKER_FILE`, `--docker-file DOCKER_FILE`: name of Dockerfile for orchestrator
- `-s SCENARIO_FILE`, `--scenario-file SCENARIO_FILE`: name of main mosaik script for orchestrator

**Optional arguments**:

- `-h`, `--help`: show the help message and exit
- `--extra-file EXTRA_FILE`: additional file to be added to the orchestrator Docker image
- `--extra-dir EXTRA_DIR`: additional directory to be added to the orchestrator Docker image
- `--result RESULT`: paths of result file or folder, i.e., file or folder produced by the simulation that should be retrieved after the simulation has finished (list of strings)

### check_sim_setup

Check if simulation setup is valid:

```
check_sim_setup [-h] [SETUP_DIR]
```

**Positional arguments**:

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)

**Optional arguments**:

- `-h`, `--help`: show the help message and exit

### build_sim_setup

Build simulation setup as preparation for running the simulation. This includes building the Docker image of the mosaik orchestrator:

```
build_sim_setup [-h] [SETUP_DIR]
```

**Positional arguments**:

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)

**Optional arguments:**

- `-h`, `--help`: show the help message and exit

### delete_sim_setup

Delete a simulation setup, including all associated Docker images and containers:

```
delete_sim_setup [-h] [SETUP_DIR]
```

**Positional arguments**:

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)

**Optional arguments:**

- `-h`, `--help`: show the help message and exit

### 3.1.2 Commands for handling simulations

#### `start_sim`

Start a new simulation:

```
start_sim [-h] [SETUP_DIR] [ID]
```

**Positional arguments:**

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)
- `ID`: simulation ID (Docker container name)

**Optional arguments:**

- `-h`, `--help`: show the help message and exit

#### `cancel_sim`

Cancel a simulation (stop simulation containers):

```
cancel_sim [-h] (--id ID | --all) [SETUP_DIR]
```

**Positional arguments**:

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)

**Optional arguments**:

- `-h`, `--help`: show the help message and exit
- `--id ID`: simulation ID (Docker container name)
- `--all`: cancel all running simulations

#### `clear_sim`

Delete containers of finished simulations:

```
clear_sim [-h] (--id ID | --all) [SETUP_DIR]
```

**Positional arguments**:

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)

**Optional arguments**:

- `-h`, `--help`: show the help message and exit

- `--id ID`: simulation ID (Docker container name)
- `--all`: remove all simulation containers

#### get_sim_status

Get status of all simulations of a mosaik-docker setup. Updates the simulation setup information about which containers are running (status *UP*) or finished(status *DOWN*) if it is not up to date:

```
get_sim_status [-h] [SETUP_DIR]
```

**Positional arguments**:

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)

**Optional arguments**:

- `-h`, `--help`: show the help message and exit

#### get_sim_results

Retrieve the results of finished simulations:

```
get_sim_results [-h] (--id ID | --all) [--overwrite] [SETUP_DIR]
```

**Positional arguments**:

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)

**Optional arguments**:

- `-h`, `--help`: show the help message and exit
- `--id ID`: simulation ID (Docker container name)
- `--all`: retrieve results from all finished simulation containers
- `--overwrite`: overwrite previously retrieved results

### 3.1.3 Utility commands

#### get_sim_ids

Get IDs of all running (status *UP*) and finished (status *DOWN*) simulations of a simulation setup:

```
get_sim_ids [-h] [SETUP_DIR]
```

**Positional arguments**:

- `SETUP_DIR`: path to simulation setup directory (default: current working directory)

**Optional arguments**:

- `-h`, `--help`: show the help message and exit

### get_sim_setup_root

Check if the specified directory (or any parent directory) contains a simulation setup configuration:

```
get_sim_setup_root [-h] [DIR]
```

**Positional arguments**:

- `DIR`: folder path (default: current working directory)

**Optional arguments**:

- `-h`, `--help`: show the help message and exit

CHAPTER 4

# Python API reference

Find detailed information about the *mosaik-docker* Python API *here*.

## 4.1 Python API reference

Package *mosaik-docker* provides the following Python methods. They can be accessed via package `mosaik_docker.cli`:

```
from mosaik_docker.cli import *
```

### 4.1.1 Methods for handling simulation setups

#### create_sim_setup

Create an empty simulation setup in a new directory.

```
create_sim_setup( name, dir = '.', id = None )
```

**Parameters**:

- `name`: name of the simulation setup (string)
- `dir`: directory to put the generated simulation setup (string, default: `'.'`)
- `id`: unique ID for the simulation setup (string, default: `None`)

**Return value**: on success, return absolute path to created simulation setup directory (string)

#### configure_sim_setup

Configure an existing simulation setup.

```
configure_sim_setup(
    setup_dir,
    docker_file,
    scenario_file,
    extra_files = [],
    extra_dirs = [],
    results = []
    )
```

**Parameters**:

- `setup_dir`: directory of the simulation setup (string, default: `'.'`)
- `docker_file`: name of the Dockerfile used for building the simulation orchestrator image (string)
- `scenario_file`: name of the mosaik scenario file (string)
- `extra_files`: additional files to be added to the simulation orchestrator image (list of strings)
- `extra_dirs`: additional directories to be added to the simulation orchestrator image (list of strings)
- `results`: list of paths of result files or folders, i.e., files or folders produced by the simulation that should be retrieved after the simulation has finished (list of strings)

**Return value**: on success, return absolute path to simulation setup config file (string)

### check_sim_setup

Check if simulation setup is valid.

```
check_sim_setup( setup_dir )
```

**Parameters**:

- `setup_dir`: path to simulation setup (string)

**Return value**: return dict with status of setup check in the following format:

```
{
    'valid': boolean # flag indicating if setup is valid
    'status': string # detailed status message
}
```

### build_sim_setup

Build simulation setup as preparation for running the simulation. This includes building the Docker image of the mosaik orchestrator.

```
build_sim_setup( setup_dir, out_stream = print ):
```

**Parameters**:

- `setup_dir`: path to simulation setup (string)
- `out_stream`: output from the build process to stderr will be piped to this stream (callable)

**Return value**: return dict with status of build process:

```
{
    'valid': flag indicating if build succeded (boolean)
    'status': detailed status message (string)
}
```

#### delete_sim_setup

Delete a simulation setup, including all associated Docker images and containers.

```
delete_sim_setup( setup_dir )
```

**Parameters**:

- `setup_dir`: path to simulation setup (string)

**Return value**: return dict with status of build process:

```
{
    'valid': flag indicating if deletion succeded (boolean)
    'status': detailed status message (string)
}
```

### 4.1.2 Methods for handling simulations

#### start_sim

Start a new simulation.

```
start_sim( setup_dir, id = None )
```

**Parameters**:

- `setup_dir`: path to simulation setup (string)
- `id`: ID of new simulation (string, default: `None`)

**Return value**: on success, return new simulation ID (int)

#### cancel_sim

Cancel a simulation (stop simulation container).

```
cancel_sim( setup_dir, id )
```

**Parameters**:

- `setup_dir`: path to simulation setup (string)
- `id`: either `'all'` or ID of running simulation container (string)

**Return value**: on success, return ID of cancelled simulation (int)

#### clear_sim

Delete containers of finished simulations.

```
clear_sim( setup_dir, id )
```

**Parameters**:

- `setup_dir`: path to simulation setup (string)
- `id`: either `'all'` or ID of simulation container to be cleared (string)

**Return value**: on success, return list of cleared simulation IDs (list of string)

#### get_sim_status

Get status of all simulations of a mosaik-docker setup. Updates the simulation setup information about which containers are running (status *UP*) or finished (status *DOWN*) if it is not up to date.

```
get_sim_status( setup_dir )
```

**Parameters**:

- `setup_dir`: path to simulation setup (string)

**Return value**: return dict with status information for all simulations of a mosaik-docker simulation setup in the following format:

```
{
    'up': { string: string } # running simulation IDs and status
    'down': { string: string } # finished simulation IDs and status
}
```

#### get_sim_results

Retrieve the results of finished simulations.

```
get_sim_results( setup_dir, id, overwrite = False )
```

**Parameters**:

- `setup_dir`: path to simulation setup (string)
- `id`: either 'all' or ID of finished simulation container (string)
- `overwrite`: overwrite previously retrieved results (boolean, default: `False`)

**Return value**: on success, return ID(s) of simulation(s) for which results have been retrieved (string)

### 4.1.3 Utility methods

Get IDs of all running (status *UP*) and finished (status *DOWN*) simulations of a simulation setup.

### get_sim_ids

```
get_sim_ids( setup_dir )
```

**Parameters**:

- `setup_dir`: path to simulation setup (string)

**Return value**: return dict with simulation IDs in the following format:

```
{
    'up': [string] # IDs of running simulations
    'down': [string] # IDs of finished simulations
}
```

### get_sim_setup_root

Check if the specified directory (or any parent directory) contains a simulation setup configuration.

```
get_sim_setup_root( dir )
```

**Parameters**:

- `dir`: directory path to check (string)

**Return value**: return the following dict:

```
{
    'valid': boolean # flag indicating if this directory (or any parent directory)␣
↪contains a simulation setup configuration
    'dir': string # directory containing a simulation setup configuration if 'valid',␣
↪otherwise empty
}
```