
Stakkr Documentation

Release 3.0

Emmanuel Dyan

Sep 24, 2017

Contents:

1	What does that do exactly ?	3
2	Examples	5
3	Indices and tables	21
	Python Module Index	23

Stakkr is a a docker recompose tool that uses docker compose to easily create / maintain a stack of services, for example for web development.

Via a configuration file you can setup the required services and let stakkr link and start everything for you.

It works only in CLI.

CHAPTER 1

What does that do exactly ?

If you have heard of Docker, you know that when you need to build a full environment with multiple services that are linked, you either have to do everything manually or use `docker-compose`. The second solution is the best *but* it implies that you need, for each environment, to change your parameters, choose your images, learn the `docker-compose` command line tool, etc ... In brief, it's not very flexible and hard to learn.

Stakkr will help you, via a very simple configuration file and a predefined list of services (that can be extended by plugins) to build a complete environment. Plus, to control it in command line. It makes use of docker easy.

Last, but not the least, it's highly configurable and each service mounts a volume to have a persistence of data. You can even, if you want, add more directives on some services (change the *php.ini* for example and choose your versions (PHP 5.3 or 5.6 or 7.1 or anything else)).

CHAPTER 2

Examples

You can combine services as you want to have :

- A **Dev LAMP stack** (Apache + MySQL 5.7 + PHP 7.2 with xdebug and xhprof) ... and if suddenly you want to test your code with PHP 7.0, change it in *conf/compose.ini*, restart, it's done !
- Or Apache 2.4 + PHP 5.6 + MongoDB for a **production environment**
- Or **only Maildev**
- Or **only PHP 5.4 + ElasticSearch**
- etc...

Installation

Docker

You must have Docker installed on your computer. Pick the right version for your OS from <https://www.docker.com/community-edition>

Prerequisites

Warning: You need to first install OS packages for Python3: `pip`, `setuptools`, `virtualenv` and (optionally) `autoenv` on your OS.

Also, to use docker for Linux as a normal user, you need to add your user to the `docker` group (see the documentation)

Example of installation of the dependencies on Ubuntu:

```
$ sudo apt-get -y install python3-pip python3-setuptools python3-virtualenv virtualenv
$ sudo pip3 install --upgrade pip
$ sudo pip3 install autoenv
```

Stakkr

There are 2 ways to intall Stakkr.

1. The easy way

Stakkr is usable as a library, it's clean, you have a very beautiful tree once installed, and it's **recommended**. You can install as many stakkr that you need. Just be careful to set different names and networks in *conf/compose.ini*

1.1 Installation under Linux

For Ubuntu, you can download Docker from : <https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/>

```
$ mkdir mydev
$ cd mydev
$ virtualenv -p /usr/bin/python3 mydev_stakkr
$ source mydev_stakkr/bin/activate
$ pip --no-cache-dir install stakkr
```

It'll run a `post_install` script that copy some templates / create base directories to work.

If you have installed autoenv, add into your `.bashrc`:

```
source `which activate.sh`
```

1.2 Installation under Windows

First install python3 from <https://www.python.org/downloads/> and docker from <https://docs.docker.com/docker-for-windows/install/>

```
> pip install virtualenv
> mkdir mydev
> cd mydev
> virtualenv venv
> venv\Scripts\activate
> pip install stakkr
```

Warning: There are known limitations under windows : First the DNS won't work and Second, *stakkr* has to create a route and change a few parameters inside MobyLinux.

1.3 Installation under MacOSX

First install python3 from <https://www.python.org/downloads/mac-osx/> (3.6 is ok) and docker from <https://docs.docker.com/docker-for-mac/install/>

```
$ mkdir mydev
$ cd mydev
$ pyenv-3.6 mydev_stakkr
$ source mydev_stakkr/bin/activate
$ pip install stakkr
```

Warning: WIP : I am currently trying to test it on Mac but it's not done yet

1.4 Development version

If you want to install the dev version, you can do the following :

```
$ pip install git+https://github.com/edyan/stakkr.git
```

2. The old way

Stakkr gets installed by cloning the github repo *not recommended if you don't develop on it.*

You can clone the repository as many times as you want as you can have multiple instances at the same time. A good practice is too have one clone for one project or one clone for projects with the same versions of PHP / MySQL / Elasticsearch, etc ...

```
$ git clone https://github.com/edyan/stakkr myenv
```

Once cloned, you can run the `install.sh` script made for Ubuntu (tested on 16.04) that will install the dependencies:

```
$ cd myenv
$ ./install.sh
```

Development

To develop, use the 2nd way to install Stakkr then :

```
$ pip install -r requirements-dev.txt
```

To generate that doc :

```
$ cd docs
$ sphinx-autobuild . _build_html
```

Configuration

Copy the file `conf/compose.ini.tpl` to `conf/compose.ini` and set the right Configuration parameters. The config validation is defined in `configspec.ini`

Main configuration parameters should be defined in the `[main]` section. Another section (`[network-block]`) has been created to define TCP ports to block for outgoing requests.

Warning: Don't use double quotes to protect your values.

Use # to comment your lines and not ;

Network and changes in general

You can define your own network in compose.ini by setting a subnet.

Warning: If you change that, run `docker-clean` which removes orphans images, stopped container, etc ...

Also, if you change any parameter such as an environment variable run a `stakkr restart --recreate` to make sure that you start from a clean environment.

Services

You can define the list of services you want to have. Each service consists of a yml file in the `services/` directory of the source code. Each container ("Virtual Machine") will have a hostname composed of the project name and the service name. To reach, for example, the elasticsearch server from a web application, and if your `project_name` = `stakkr` uses `stakkr_elasticsearch` or to connect to mysql use `stakkr_mysql`. The service names also works (*elasticsearch* and *mysql*)

```
# Comma separated list of services to start
# Valid values: adminer / apache / elasticsearch / elasticsearch-old / mailcatcher / ↵
↵maildev
# mongo / mysql / php / phpmyadmin / python / redis / xhgui
services=apache,php,mysql
```

A service can launch a post-start script that has the same name with an `.sh` extension (example: `services/mysql.sh`).

Special case of xhgui service

To be able to profile your script, add the service `xhgui` and read the [documentation](#)

Other useful parameters

Project name (will be used as container's prefix). It should be different for each project.

```
# Change Machines names only if you need it
project_name=stakkr
```

PHP Version :

```
# Set your PHP version from 5.3 to 7.0 (5.6 by default)
php.version=7.0
```

MySQL Password if mysql is defined in the services list:

```
# Password set on first start. Once the data exist won't be changed
mysql.root_password=changeme
```

Memory assigned to the VMS:

```
apache.ram=512M
elasticsearch.ram=512M
mysql.ram=512M
php.ram=512M
```

Port Blocking: by default, we can block ports only for the PHP container (as iptables is installed). Define in a list what port you want to **block for OUTPUT TCP requests**. That has been done to avoid mistakes such as using a production database and send a lot of emails ...

```
[network-block]
php=25,465,587
```

Files location

Public Files

- All files served by the web server are located into `www/`

Services Data

- MySQL data is into `data/mysql`
- Mongo data is into `data/mongo`
- ElasticSearch data is into `data/elasticsearch`
- Redis data is into `data/redis`

Logs

- Logs for Apache and PHP are located into `logs/`
- Logs for MySQL are located into `data/mysql/` (slow and error).

Configuration

- If you need to override the PHP configuration you can put a file in `conf/php-fpm-override` with a `.conf` extension. The format is the fpm configuration files one. Example: `php_value[memory_limit] = 127M`.
- If you need to override the mysql configuration you can put a file in `conf/mysql-override` with a `.cnf` extension.

Add binaries

You can add binaries (such as `phpunit`) that will automatically be available from the `PATH` by putting it to `home/www-data/bin/`

Important: You can use `home/www-data` to put everything you need to keep: your shell parameters in `.bashrc`, your ssh keys/config into `.ssh`, etc.

Usage

Before running any command

Important:

You have to be in a virtual environment. To verify that, check that your prompt starts with something like (xyz_stakkr)

If you have autoenv, and if you kept the name of the virtualenv as described above, just enter the directory, and it'll be automatically activated. Else:

```
$ source ${PWD##*/}_stakkr/bin/activate
```

To leave that environment:

```
$ deactivate
```

Get Help

To get a list of commands do `stakkr --help` and to get help for a specific command : `stakkr start --help`

CLI Reference

Docker Commands

docker-clean

Clean Docker containers, images, volumes and networks that are not in use

```
docker-clean [OPTIONS]
```

Options

-f, --force

Do it

-v, --verbose

Display more information about what is removed

stakkr-compose

Wrapper for docker-compose

```
stakkr-compose [OPTIONS] [COMMAND]...
```

Options

-c, --config <config>
Override the conf/compose.ini

Arguments

COMMAND
Optional argument(s)

Stakkr Commands

stakkr

Main CLI Tool that easily create / maintain a stack of services, for example for web development.

Read the configuration file and setup the required services by linking and managing everything for you.

```
stakkr [OPTIONS] COMMAND [ARGS]...
```

Options

--version
Show the version and exit.

-c, --config <config>
Change the configuration file

-d, --debug, --no-debug

-v, --verbose

console

Enter a container to perform direct actions such as install packages, run commands, etc.

```
stakkr console [OPTIONS] CONTAINER
```

Options

-u, --user <user>
User's name. Valid choices : www-data or root

-t, --tty, --no-tty
Use a TTY

Arguments

CONTAINER
Required argument

dns

Start or Stop the DNS forwarder. Useful to access your containers directly by their names. Does not work under Windows as we can't mount `/etc/resolv.conf`.

Valid values for ACTION : 'start' or 'stop'

```
stakkr dns [OPTIONS] ACTION
```

Arguments

ACTION

Required argument

exec

Execute a command into a container.

Examples:

- `stakkr -v exec mysql mysqldump -p'$MYSQL_ROOT_PASSWORD' mydb > /tmp/backup.sql`
- `stakkr exec php php -v` : Execute the php binary in the php container with option -v
- `stakkr exec apache service apache2 restart`

```
stakkr exec [OPTIONS] CONTAINER COMMAND...
```

Options

-u, --user <user>

User's name. Be careful, each container have its own users.

-t, --tty, --no-tty

Use a TTY

Arguments

CONTAINER

Required argument

COMMAND

Required argument(s)

mysql

stakkr mysql is a wrapper for the mysql binary located in the mysql service.

You can run any mysql command as root, such as :

- `stakkr mysql -e "CREATE DATABASE mydb"` to create a DB from outside

- `stakkr mysql` to enter the mysql console
- `cat myfile.sql | stakkr mysql --no-tty mydb` to import a file from outside to mysql

For scripts, you must use the relative path.

```
stakkr mysql [OPTIONS] [COMMAND]...
```

Options

-t, --tty, --no-tty
Use a TTY

Arguments

COMMAND
Optional argument(s)

refresh-plugins

Required to be launched if you install a new plugin

```
stakkr refresh-plugins [OPTIONS]
```

restart

Restart all containers

```
stakkr restart [OPTIONS]
```

Options

-p, --pull
Force a pull of the latest images versions

-r, --recreate
Recreate all containers

start

Start containers defined in compose.ini

```
stakkr start [OPTIONS]
```

Options

- p, --pull**
Force a pull of the latest images versions
- r, --recreate**
Recreate all containers

status

Display a list of running containers

```
stakkr status [OPTIONS]
```

stop

Stop the services

```
stakkr stop [OPTIONS]
```

Stakkr Init

stakkr-init

Initialize for the first time stakkr by copying templates and directory structure

```
stakkr-init [OPTIONS]
```

Options

- f, --force**
Force recreate directories structure

Plugins development

Write a plugin

To write a plugin you need to create a folder in the `plugins/` directory that contains your commands.

Warning: Each directory must contain a `setup.py` to be installed as a plugin. Check the following link to have more info about how to build a plugin: <https://github.com/click-contrib/click-plugins/tree/master/example>

Of course you can use any module included in stakkr during your developments (`click`, `clint`, `stakkr.command`, `stakkr.docker`, `stakkr.package_utils`, etc...).

Example

You want to build a simple command that says “Hello”. It’ll be called `_sayhello_`. You need to create two files in a `sayhello` directory.

- `plugins/sayhello/setup.py`

```
from setuptools import setup

setup(
    name='StakkrSayHello',
    version='1.0',
    packages=['sayhello'],
    entry_points="""
        [stakkr.plugins]
        sayhello=sayhello.core:hi
    """)
```

- And `plugins/sayhello/sayhello/core.py`

```
import click

@click.command(help="Example")
def hi():
    print('Hi!')
```

Once your plugin has been installed you need to run:

```
$ stakkr refresh-plugins
$ stakkr hi
```

Install a plugin

To install a plugin

```
$ cd plugins/
$ git clone https://github.com/xyz/stakkr-myplugin myplugin
$ stakkr refresh-plugins
```

You can, for example install composer plugin:

```
$ cd plugins/
$ git clone https://github.com/edyan/stakkr-composer composer
$ stakkr refresh-plugins
$ cd ../www
$ stakkr composer
```

Define services in your plugins

By creating a `services/` directory you can either override or create new services with your plugins. Example: `plugins/myplugin/services/mysql.yml` will override the default mysql service while `plugins/myplugin/services/nginx.yml` will define a new service.

Each service added by a plugin must be added in `compose.ini` to be started.

Example of a service:

```
version: '2.2'

services:
  nginx:
    image: nginx
    container_name: ${COMPOSE_PROJECT_NAME}_nginx
    hostname: ${COMPOSE_PROJECT_NAME}_nginx
    networks: [stakkr]
```

List of existing plugins

- **stakkr-composer** : Download and run composer
- **stakkr-sugarcli** : Download and run sugarcli
- **stakkr-phing** : Download and run Phing

Custom Services

Overview

If you need a specific service that is not included in stakkr by default, you can add a yml file into `services/` directory.

Write a Service

A stakkr service respects the `docker-compose` standard, plus a few customizations.

Some rules:

- The yml file must be named with the same name than the service
- That name will help to define the name of the service in `conf/compose.ini`
- You are free to add everything you want to `conf/compose.ini`
- A configuration parameter such as `php.ram` generates an environment variable that looks like `DOCKER_PHP_RAM`.

Example

Let's make an nginx service. The file will be located into `services/` as `nginx.yml`.

```
version: '2'

services:
  nginx:
    image: nginx:${DOCKER_NGINX_VERSION}
    mem_limit: ${DOCKER_NGINX_RAM}
    container_name: ${COMPOSE_PROJECT_NAME}_nginx
    hostname: ${COMPOSE_PROJECT_NAME}_nginx
    networks: [stakkr]
```

```
ports:
  - "8080:80"
```

Now in `conf/compose.ini`:

```
services=nginx
nginx.version=1.13-alpine
nginx.ram=256M
```

Restart:

```
$ stakkr restart --recreate
$ stakkr status
```

To run a command, use the standard `exec` wrapper:

```
$ stakkr exec nginx cat /etc/nginx/nginx.conf
```

Stakkr's code structure

Stakkr works with a few modules / classes:

Module `stakkr.actions`

Stakkr main controller. Used by the CLI to do all its actions

```
class stakkr.actions.StakkrActions (base_dir: str, ctx: dict)
    Main class that does actions asked in the cli

    console (container: str, user: str, tty: bool)
        Enter a container (stakkr allows only apache / php and mysql)

    exec_cmd (container: str, user: str, args: tuple, tty: bool)
        Run a command from outside to any container. Wrapped into /bin/sh

    get_services_ports ()
        Once started, stakkr displays a message with the list of launched containers.

    start (pull: bool, recreate: bool)
        If not started, start the containers defined in config

    status ()
        Returns a nice table with the list of started containers

    stop ()
        If started, stop the containers defined in config. Else throw an error
```

Module `stakkr.command`

A command wrapper to get a live output displayed. Useful when you need to write a plugin that outputs some progress or info.

```
stakkr.command.launch_cmd_displays_output (cmd: list, print_msg: bool = True, print_err: bool = True, err_to_out: bool = False)
    Launch a command and displays conditionnaly messages and / or errors
```

`stakkr.command.verbose` (*display: bool, message: str*)
Display a message if verbose is On

Module `stakkr.configreader`

Simple Config Reader

```
class stakkr.configreader.Config (config_file: str = None)
    Parser of Stakkr. Set default values and validate conf/compose.ini with conf/configspec.ini

    display_errors ()
        Display errors in STDOUT

    read ()
        Read the default values and overridden ones
```

Module `stakkr.docker_actions`

Docker functions to get info about containers

```
stakkr.docker_actions.add_container_to_network (container: str, network: str)
    Attach a container to a network

stakkr.docker_actions.block_ct_ports (service: str, ports: list, project_name: str) → tuple
    Run iptables commands to block a list of port on a specific container

stakkr.docker_actions.check_cts_are_running (project_name: str)
    Throws an error if cts are not running

stakkr.docker_actions.container_running (container: str)
    Returns True if the container is running else False

stakkr.docker_actions.create_network (network: str)
    Create a Network

stakkr.docker_actions.get_api_client ()
    Returns the API client or initialize it

stakkr.docker_actions.get_client ()
    Returns the client or initialize it

stakkr.docker_actions.get_ct_item (compose_name: str, item_name: str)
    Get a value from a container, such as name or IP

stakkr.docker_actions.get_ct_name (container: str)
    Returns the system name of a container, generated by docker-compose

stakkr.docker_actions.get_running_containers (project_name: str) → tuple
    Get the number of running containers and theirs details for the current stakkr instance

stakkr.docker_actions.get_running_containers_name (project_name: str) → list
    Get a list of compose names of running containers for the current stakkr instance

stakkr.docker_actions.get_subnet (project_name: str)
    Find the subnet of the current project

stakkr.docker_actions.get_switch_ip ()
    find the main docker daemon IP to add routes

stakkr.docker_actions.guess_shell (container: str) → str
    By searching for binaries, guess what could be the primary shell available
```

`stakkr.docker_actions.network_exists` (*network: str*)
True if a network exists in docker, else False

Module `stakkr.package_utils`

Gives useful information about the current virtualenv, files locations if stakkr is installed as a package or directly cloned

`stakkr.package_utils.get_dir` (*dirname: str*)
Detects if stakkr is a package or a clone and gives the right path for a directory

`stakkr.package_utils.get_file` (*dirname: str, filename: str*)
Detects if stakkr is a package or a clone and gives the right path for a file

`stakkr.package_utils.get_venv_basedir` ()
Returns the base directory of the virtualenv, useful to read configuration and plugins

Module `stakkr.plugins`

Module used by `setup.py` to find plugins to load with click

`stakkr.plugins.add_plugins` ()
Read the plugins directory, get the subfolders from it and look for `.py` files

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`

S

- `stakkr.actions`, [17](#)
- `stakkr.command`, [17](#)
- `stakkr.configreader`, [18](#)
- `stakkr.docker_actions`, [18](#)
- `stakkr.package_utils`, [19](#)
- `stakkr.plugins`, [19](#)

Symbols

-version
 stakkr command line option, 11
 -c, --config <config>
 stakkr command line option, 11
 stakkr-compose command line option, 11
 -d, --debug, --no-debug
 stakkr command line option, 11
 -f, --force
 docker-clean command line option, 10
 stakkr-init command line option, 14
 -p, --pull
 stakkr-restart command line option, 13
 stakkr-start command line option, 14
 -r, --recreate
 stakkr-restart command line option, 13
 stakkr-start command line option, 14
 -t, --tty, --no-tty
 stakkr-console command line option, 11
 stakkr-exec command line option, 12
 stakkr-mysql command line option, 13
 -u, --user <user>
 stakkr-console command line option, 11
 stakkr-exec command line option, 12
 -v, --verbose
 docker-clean command line option, 10
 stakkr command line option, 11

A

ACTION

 stakkr-dns command line option, 12
 add_container_to_network() (in module
 stakkr.docker_actions), 18
 add_plugins() (in module stakkr.plugins), 19

B

block_ct_ports() (in module stakkr.docker_actions), 18

C

check_cts_are_running() (in module
 stakkr.docker_actions), 18

COMMAND

 stakkr-compose command line option, 11
 stakkr-exec command line option, 12
 stakkr-mysql command line option, 13

Config (class in stakkr.configreader), 18

console() (stakkr.actions.StakkrActions method), 17

CONTAINER

 stakkr-console command line option, 11
 stakkr-exec command line option, 12
 container_running() (in module stakkr.docker_actions),
 18
 create_network() (in module stakkr.docker_actions), 18

D

display_errors() (stakkr.configreader.Config method), 18

docker-clean command line option

 -f, --force, 10
 -v, --verbose, 10

E

exec_cmd() (stakkr.actions.StakkrActions method), 17

G

get_api_client() (in module stakkr.docker_actions), 18
 get_client() (in module stakkr.docker_actions), 18
 get_ct_item() (in module stakkr.docker_actions), 18
 get_ct_name() (in module stakkr.docker_actions), 18
 get_dir() (in module stakkr.package_utils), 19
 get_file() (in module stakkr.package_utils), 19
 get_running_containers() (in module
 stakkr.docker_actions), 18
 get_running_containers_name() (in module
 stakkr.docker_actions), 18
 get_services_ports() (stakkr.actions.StakkrActions
 method), 17
 get_subnet() (in module stakkr.docker_actions), 18

get_switch_ip() (in module stakkr.docker_actions), 18
get_venv_basedir() (in module stakkr.package_utils), 19
guess_shell() (in module stakkr.docker_actions), 18

L

launch_cmd_displays_output() (in module stakkr.command), 17

N

network_exists() (in module stakkr.docker_actions), 19

R

read() (stakkr.configreader.Config method), 18

S

stakkr command line option

- version, 11
- c, --config <config>, 11
- d, --debug, --no-debug, 11
- v, --verbose, 11

stakkr-compose command line option

- c, --config <config>, 11
- COMMAND, 11

stakkr-console command line option

- t, --tty, --no-tty, 11
- u, --user <user>, 11
- CONTAINER, 11

stakkr-dns command line option

- ACTION, 12

stakkr-exec command line option

- t, --tty, --no-tty, 12
- u, --user <user>, 12
- COMMAND, 12
- CONTAINER, 12

stakkr-init command line option

- f, --force, 14

stakkr-mysql command line option

- t, --tty, --no-tty, 13
- COMMAND, 13

stakkr-restart command line option

- p, --pull, 13
- r, --recreate, 13

stakkr-start command line option

- p, --pull, 14
- r, --recreate, 14

stakkr.actions (module), 17

stakkr.command (module), 17

stakkr.configreader (module), 18

stakkr.docker_actions (module), 18

stakkr.package_utils (module), 19

stakkr.plugins (module), 19

StakkrActions (class in stakkr.actions), 17

start() (stakkr.actions.StakkrActions method), 17

status() (stakkr.actions.StakkrActions method), 17

stop() (stakkr.actions.StakkrActions method), 17

V

verbose() (in module stakkr.command), 17