PhpZone Docker Documentation

Release 0.2

Jakub Zapletal

August 21, 2015

CHAPTER 1

Getting Started

1.1 Requirements

PhpZone requires PHP 5.3 or higher.

1.2 Installation

Installation is provided via Composer, if you don't have it, do install:

```
$ curl -s https://getcomposer.org/installer | php
```

then PhpZone Docker Compose can be added into your dependencies by:

```
$ composer require --dev phpzone/docker 0.2.*
```

or add it manually into your `composer.json`:

```
{
    "required-dev": {
        "phpzone/docker": "0.2.*"
    }
}
```

1.3 Configuration file

If the configuration file doesn't exist yet, you can find more information in PhpZone - Getting Started

1.3.1 Registration of the extension

Registration is provided by a simple definition of full name of the class (namespace included):

```
extensions:
    PhpZone\Docker\DockerCompose: ~
```

Note: This extension is a command builder with definitions within its values. This means that only the registration without any values would have no effect.

1.3.2 Creating of commands

As mentioned in the PhpZone documentation, each extension gets its configuration via values which are defined during its registration. PhpZone Docker Compose expects to get an array of required commands. Each command is defined with it's name as the key and the Docker command to be run as the value:

```
extensions:
    PhpZone\Docker\DockerCompose:
        command_name:
            command: up
```

Now, when we would run:

```
$ vendor/bin/phpzone command_name
```

the command `docker-compose up` would be run.

CHAPTER 2

Definitions For Command

Commands can contain following definitions:

Name	Type	Default	Required
build	array		No
command	string	up	No
description	string	null	No
enable	boolean	true	No
file	string	docker-compose.yml	No
help	string	null	No
name	string	*current directory name*	No
parent	string	null	No
rm	array		No
scale	array		No
up	array		No
verbose	boolean	false	No

Example with default values:

```
extensions:
    PhpZone\Docker\DockerCompose:
        command_name:
            build:
                no_cache: false
            command: up
            description: ~
            enable: true
            file: ~
            help: ~
            name: ~  # current directory name
            parent: ~
            rm:
                force: false
            scale: ~
            up:
                daemon:      false
                no_recreate: false
                no_build:    false
            verbose: false
```

Note: The order of the definitions is not important.

Note: Not required definitions don't need to be set.

2.1 build

Type	Default	Required
array		No

Extended options in case of the definition `command: build`.

Example with default values:

```
extensions:
    PhpZone\Docker\DockerCompose:
        command_name:
            build:
                no_cache: false
```

2.1.1 no_cache

Type	Default	Required
boolean	false	No

Do not use cache when building the image.

2.2 command

Type	Default	Required
string	up	No

Specifies which command will be run against services. Available commands are: `build`, `kill`, `logs`, `ps`, `pull`, `rm`, `scale`, `start`, `stop` and `up`. For more information read official documentation of Docker Compose.

2.3 description

Type	Default	Required
string	null	No

The description of a command will be displayed when a developer would run the command `list` or without any command.

2.4 enable

Type	Default	Required
boolean	true	No

All defined commands are enabled by default. Sometimes can be useful to disable a command without its removal.

2.5 file

Type	Default	Required
string	docker-compose.yml	No

Specifies an alternate Compose yaml file. Official documentation of Docker Compose

2.6 help

Type	Default	Required
string	null	No

The help of a command will be displayed when a developer would run the command `help`.

2.7 name

Type	Default Required	
string	*current directory name*	No

Specifies an alternate project name. Official documentation of Docker Compose

2.8 parent

Type	Default	Required
string	null	No

It can help you to define more commands related to the same definitions, so it can help to avoid duplications. The value is defined as `parent: command_name`.

Example:

```
extensions:
    PhpZone\Docker\DockerCompose:
        command_name_1:
            command: up
            name:    myproject
        command_name_2:
            command: stop
            parent:  command_name_1
```

If you run:

```
$ vendor/bin/phpzone comand_name_2
```

This will compose `docker-compose -p myproject stop` and execute it.

2.9 rm

Type	Default	Required
array		No

Extended options in case of the definition `command: rm`.

Example with default values:

```
extensions:
    PhpZone\Docker\DockerCompose:
        command_name:
            rm:
                force: false
```

2.9.1 force

Type	Default	Required
boolean	false	No

Don't ask to confirm removal.

2.10 scale

Type	Default	Required
array		No

Extended options in case of the definition `command: scale`. Numbers are specified in the form `service_name: integer`.

Example:

```
extensions:
    PhpZone\Docker\DockerCompose:
        command_name:
            scale:
                service_name_1: 3
                service_name_2: 2
```

2.11 up

Type	Default	Required
array		No

Extended options in case of the definition `command: up`.

Example with default values:

```
extensions:
    PhpZone\Docker\DockerCompose:
        command_name:
            up:
                daemon:      false
                no_recreate: false
                no_build:    false
```

2.11.1 daemon

Type	Default	Required
boolean	false	No

Detached mode: Run containers in the background, print new container names.

2.11.2 no_recreate

Type	Default	Required
boolean	false	No

If containers already exist, don’t recreate them.

2.11.3 no_build

Type	Default	Required
boolean	false	No

Don’t build an image, even if it’s missing.

2.12 verbose

Type	Default	Required
boolean	false	No

Show more output. Official documentation of Docker Compose

CHAPTER 3

Options For Command

All built commands have their definitions of how they are executed, but sometimes it's useful to have an option to rewrite a defined parameter or extend a functionality. This is provided by options below.

Options are extended attributes which can be set either before command name or after command name, so both following examples are valid:

```
$ vendor/bin/phpzone <OPTION> <COMMAND>
```

```
$ vendor/bin/phpzone <COMMAND> <OPTION>
```

Tip: All available options can be displayed by:

```
$ vendor/bin/phpzone <COMMAND> --help
```

3.1 –build

Overwrites defined command by `build`. Command definition

3.2 –kill

Overwrites defined command by `kill`. Command definition

3.3 –logs

Overwrites defined command by `logs`. Command definition

3.4 –ps

Overwrites defined command by `ps`. Command definition

3.5 –pull

Overwrites defined command by `pull`. Command definition

3.6 –rm

Overwrites defined command by `rm`. Command definition

3.7 –scale

Overwrites defined command by `scale`. Command definition

3.8 –start

Overwrites defined command by `start`. Command definition

3.9 –stop

Overwrites defined command by `stop`. Command definition

3.10 –up

Overwrites defined command by `up`. Command definition

A Docker command builder configured by YAML, based on PhpZone. Its primary purpose is to provide a simple way to define commands for running Docker containers/instances which could be used in daily workflow of every developer. With this plugin, you no longer need to have advanced knowledge about Docker, as you can add store simple commands to your configuration. This is also useful for advanced users who want to store ready-made commands.

Attention: This tool is only a configurator and executor of Docker commands, applications Docker and **‘Docker Compose‘_** are not included.

Basic Usage

An example speaks a hundred words so let's go through one.

The configuration file below is used for the development of this extension:

Create a `phpzone.yml` file in the root of a project:

```
extensions:
    PhpZone\Docker\DockerCompose: # register an extension with a configuration
        db:
            description: Run DB which can be used for running tests
            name: myproject
            file: docker-compose.yml
            command: up
```

and run:

```
$ vendor/bin/phpzone db
```

This will compose a proper Docker Compose command `docker-compose -f docker-compose.yml -p myproject up` and execute it.