Linux For Composer Documentation

Release 2.0.9

Foreach Code Factory

Jul 10, 2020

Contents

Linux for PHP Home Page

CHAPTER 1

Configure and run Linux for PHP containers without ever touching the Docker command line!

You can create a custom configuration for each PHP project you have and launch the Linux for PHP containers directly from within your projects' working directories!

1.1 Installation

1.1.1 Prerequisites

Linux for Composer runs in the default Docker environment of each supported platform (operating system).

- **All platforms:**
  - Docker (https://www.docker.com/)
  - Composer (https://getcomposer.org/)
  - Git (https://git-scm.com/)
  - cURL (https://curl.haxx.se/)
- Unix/Mac/Linux: Bash,
- Windows 10: PowerShell,
- Windows 8: Bash for Windows (Docker Toolbox).

1.1.2 Installation using Composer

To install the **Linux for Composer** package, you can simply run the following commands:

```
$ composer require --dev linuxforphp/linuxforcomposer
$ php vendor/bin/linuxforcomposer.phar
```

Note: On Windows, please use the PHAR file in the `vendor/linuxforphp/linuxforcomposer/bin` folder.

You can install **Linux for Composer** for your entire system by copying the binary in a folder that is in your **PATH**:

```
$ cp vendor/linuxforphp/linuxforcomposer/bin/linuxforcomposer.phar /usr/local/bin/
↪linuxforcomposer
```

You would then be able to invoke the binary directly from within the working directories of your PHP projects:

```
$ cd /my/favorite/php/project
$ linuxforcomposer docker:run start
```

Once installed, you will now be able to configure the `linuxforcomposer.json` file according to the specific needs of your project.

1.2 Configuration

To obtain a sample configuration file, please run this command in a **Composer**-enabled project's folder, once **Linux for Composer** has been installed:

```
$ php vendor/bin/linuxforcomposer.phar docker:run start
```

You will automatically initialize the project with the following default configuration:

```
{
    "name": "linuxforphp/linuxforcomposer",
    "description": "A Composer interface to run 'Linux for PHP' Docker containers,␣
↪Dockerfiles or docker-compose files.",
    "single": {
        "image": {
            "linuxforcomposer": {
                "php-versions": [
                    "7.4",
                    "7.3",
                    "7.2"
                ],
                "script": [
                    "echo '<?php phpinfo();' > /srv/www/index.php",
                    "lfphp --mysql --phpfpm --apache"
                ],
                "thread-safe": "false"
            },
            "dockerfile": {
                "url": "",
                "container-name": "",
                "username": "",
                "token": ""
            }
        },
        "containers": {
            "modes": {
                "mode1": "detached",
                "mode2": "interactive",
```

(continues on next page)

(continued from previous page)

```
                "mode3": "tty"
            },
            "ports": {
                "port1": [
                    "7474:80",
                    "7373:80",
                    "7272:80"
                ],
                "port2": [
                    "13306:3306",
                    "13307:3306",
                    "13308:3306"
                ]
            },
            "volumes": {
                "volume1": "",
                "volume2": ""
            },
            "persist-data": {
                "mount": "false",
                "root-name": "",
                "directories": {
                    "directory1": "",
                    "directory2": "",
                    "directory3": ""
                }
            }
        }
    },
    "docker-compose": {
        "url": "",
        "username": "",
        "token": ""
    },
    "lfphp-cloud": {
        "account": "",
        "username": "",
        "token": ""
    }
}
```

Linux for Composer has three main modes:

- **Single** mode
- **Docker Compose** mode
- **Linux for PHP Cloud** mode

Only one of these modes can be used at one time.

When using the `single` mode, one must configure the image that should be used or built, and the containers that should be spun up.

The image can be configured through the `image` setting, and by using the `linuxforcomposer` standard mode, which uses **Linux for PHP** images in the background, or by using the `dockerfile` mode, which uses any of the images that you can find in the **Docker Hub** repositories. The `dockerfile` mode has precedence over the `linuxforcomposer` mode.

Note: For more information on configuring the `linuxforcomposer` mode, please see the *Linux for Composer Mode* section.

Since the `dockerfile` mode has precedence over the `linuxforcomposer` mode, one can keep all of the `linuxforcomposer` configurations intact by simply adding a `dockerfile` configuration, which will cause the `linuxforcomposer` configurations to be totally ignored. This is useful when spinning up an ad hoc image to quickly test something in a project's code base.

Note: For more information on configuring the `dockerfile` mode, please see the *Dockerfile Mode* section.

Once the image is configured, the containers must be configured by using the `containers` setting. In this section, it is possible to configure the modes, ports, volumes and mount points for each container.

Note: For more information on configuring the `containers` setting, please see the *Containers* section.

When not using `single` mode, but when using the `docker-compose` mode instead, one must give the URL of a Git repository which contains a valid `docker-compose.yml` file in its root folder. Private repositories are also supported, but require that a `username` and an access `token` be given in this section of the `linuxforcomposer.json` file.

Note: For more information on configuring the `docker-compose` mode, please see the *Docker Compose Mode* section.

Finally, when not using either of the previous modes (`single` or `docker-compose`), but when using the `lfphp-cloud` mode in their place, it is possible to set up an automatic deployment of a project to the **Linux for PHP Cloud**, by configuring the `account` name, the `username`, and the public access `token` to a valid account.

Note: For more information on configuring the `lfphp-cloud` mode, please see the *Linux for PHP Cloud Mode* section.

For more details on how to get a **Linux for PHP Cloud** account, please see the Linux for PHP Cloud Services website.

1.2.1 Single Mode

In `single` mode, **Linux for Composer** will either use a **Linux for PHP** image, or an image that will be built using a Dockerfile. Once the image is ready, **Linux for Composer** will spin up one or more containers according to the options given in the *Containers* setting.

1.2.1.1 Image

The image section configures **Linux for Composer** to use and/or build an image. One must choose between the *Linux for Composer Mode* or *Dockerfile Mode* mode.

Note: The `dockerfile` mode has precedence over the `linuxforcomposer` mode.

1.2.1.1.1 Linux for Composer Mode

The main configuration settings for the `linuxforcomposer` mode are:

- *PHP Versions*
- *Scripts*
- *Thread-Safety*

PHP Versions

`php-versions` (Required - Default: none)

A list of the available pre-compiled versions can be found in the Linux for PHP repository on Docker Hub.

If many versions are chosen at once, **Linux for Composer** will start a detached container for each chosen version.

If you wish to obtain an interactive shell, enter `/bin/bash` in the script section (see *Scripts*) and do not ask for the 'detached' mode in the modes section (see *Modes*).

Finally, if you enter a version number like `8.0` (without the 'dev' part), **Linux for Composer** will COMPILE the latest version from source!!! Now, that's really bleeding edge, isn't it?

Scripts

`script` (Optional - Default: 'lfphp')

You can enter any command that you wish to execute as soon as the **Linux for PHP** container has finished starting. The most common ones are 'lfphp' and '/bin/bash'. But, you could also execute a PHP script directly or launch one of the recipes from the Linux for PHP documentation. You may enter as many commands as you need, as long as you enter one command per line of the script setting.

For example, to install **Drupal** automatically, you could enter:

`"lfphp-get cms drupal testapp"`

Another example would be to install **Laravel**:

`"lfphp-get php-frameworks laravel testapp"`

And, then, to start the LAMPP stack only:

`"lfphp --mysql --phpfpm --apache"`

Thread-Safety

`thread-safe` (Optional - Default: 'false')

It is possible to run a Zend thread-safe ('true') or a non-thread safe ('false') version of PHP.

1.2.1.1.2 Dockerfile Mode

`dockerfile` (Optional - Default: none)

When configuring the `dockerfile` mode, one must give the `url` of the Dockerfile that is to be used, and a name (`container-name`) to the image and its subsequently-created container. The file's URL can be local (path) or remote (http/https protocols only). If a remote Dockerfile requires authentication, it is possible to add a `username` and an access `token` to access a private repository, for example.

```
[...]

"dockerfile": {
    "url": "Dockerfile",
    "container-name": "specialproject",
    "username": "",
    "token": ""
}

[...]
```

Or,

```
[...]

"dockerfile": {
    "url": "https://example.com/repo/Dockerfile",
    "container-name": "specialproject2",
    "username": "user1",
    "token": "roviquerhoqiuerhvoqierbvoi"
}

[...]
```

Note: Please make sure cURL and Git are available on your system when trying to access remote files.

1.2.1.2 Containers

The main configuration settings for the `containers` section are:

- *Modes*
- *Ports*
- *Volumes*
- *Persist Data*

1.2.1.2.1 Modes

`modes` (Optional - Default: detached mode)

There are three possible modes when running Docker containers with **Linux for Composer**:

- Detached
- Interactive
- TTY

Whenever, you ask for the detached mode, it will take precedence over any other mode that you ask for in the `linuxforcomposer.json` file.

Note: For more information on Docker modes, please read the Docker documentation.

1.2.1.2.2 Ports

`ports` (Optional - Default: none)

You can share ports from the host system with your containers.

If you enter many port mappings for each shared port, **Linux for Composer** will share each mapping with one container in the order they were given. For example, 'port1' contains two mappings (8181:80 and 8282:80) and so does 'port2' (13306:3306 and 13307:3306). The first element of each mapping (8181:80 and 13306:3306) will be given to container 1, which corresponds to the first given PHP version in the `php-versions` section (see *PHP Versions*). The second element of each mapping (8282:80 and 13307:3306) will be given to container 2.

1.2.1.2.3 Volumes

`volumes` (Optional - Default: none)

You can share volumes between the host and your containers.

Note: Each volume will be shared with each and every container.

Linux/Unix/Mac users can insert Bash environment variables in this part of the JSON file. For example, you can share your current working directory with your containers by entering: "${PWD}/:/srv/www". This will make your working directory available to the web server inside the Linux for PHP container.

On Windows 10 (PowerShell), please share the volume by using the following format:

`"c:/Users/test:/srv/test"`

On Windows 8 (Bash), please use the following format:

`"/c/Users/test:/srv/test"`

Note: Windows users must make sure to turn volume sharing on in the Docker settings.

1.2.1.2.4 Persist Data

`persist-data` (Optional - Default: false)

You can persist data by using **Docker** volumes and mounting them inside a container or sharing them between containers.

To mount **Docker** volumes to persist data from inside the container, one must set the `mount` setting to `true`, give a root name to the mounted volumes (we recommend setting it to the unique name of the project), and giving the names of the container's directories that should be persisted. For example, one could persist the container's '/srv' folder like so:

```
[...]

"persist-data": {
    "mount": "true",
    "root-name": "unique_name_of_my_project",
    "directories": {
        "directory1": "/srv",
        "directory2": "",
        "directory3": ""
    }
}

[...]
```

This will instruct **Linux for Composer** to create a **Docker** volume with the name `unique_name_of_my_project_srv` and to share it with the container(s) created in the *Linux for Composer Mode*, or the container created in the *Dockerfile Mode*.

Upon creation of the volume, **Linux for Composer** will sync the new volume with the data that it will find in the designated directory.

Note: Windows containers are still NOT supported as of version 2.0.0 of Linux for Composer.

1.2.2 Docker Compose Mode

`docker-compose` (Optional - Default: none)

When configuring the `docker-compose` mode, one must give the `url` of the Git repository that has the main `docker-compose.yml` file in its root folder. The repository's URL can be local (path) or remote (Git supported protocols only). If the remote repository requires authentication, it is possible to add a `username` and a `token` to access the repository.

```
[...]

"docker-compose": {
    "url": "asclinux-docker-compose",
    "username": "",
    "token": ""
},

[...]
```

Or,

```
[...]

"docker-compose": {
    "url": "https://github.com/andrewscaya/asclinux-docker-compose",
    "username": "",
    "token": ""
},

[...]
```

Note: Please make sure cURL and Git are available on your system when trying to access remote files.

1.2.3 Linux for PHP Cloud Mode

`lfphp-cloud` (Optional - Default: none)

When configuring the `lfphp-cloud` mode, one must give the name of the `account`, the `username` and the public access `token` of the **Linux for PHP Cloud Services** for the account that should to be used.

```
[...]

"lfphp-cloud": {
    "account": "johnsmithexamplecom1",
    "username": "john.smith@example.com",
    "token": "rnvaernlaiurnaliurnfgalriunvaernveiruneirug"
}

[...]
```

Note: Not all Linux for Composer settings are available on all Linux for PHP Cloud hosting plans. For more information, please see the Linux for PHP Cloud Services website.

1.3 Usage

1.3.1 linuxforcomposer docker:run start

Once you are done modifying the JSON file, you can start the container or containers by issuing the following command:

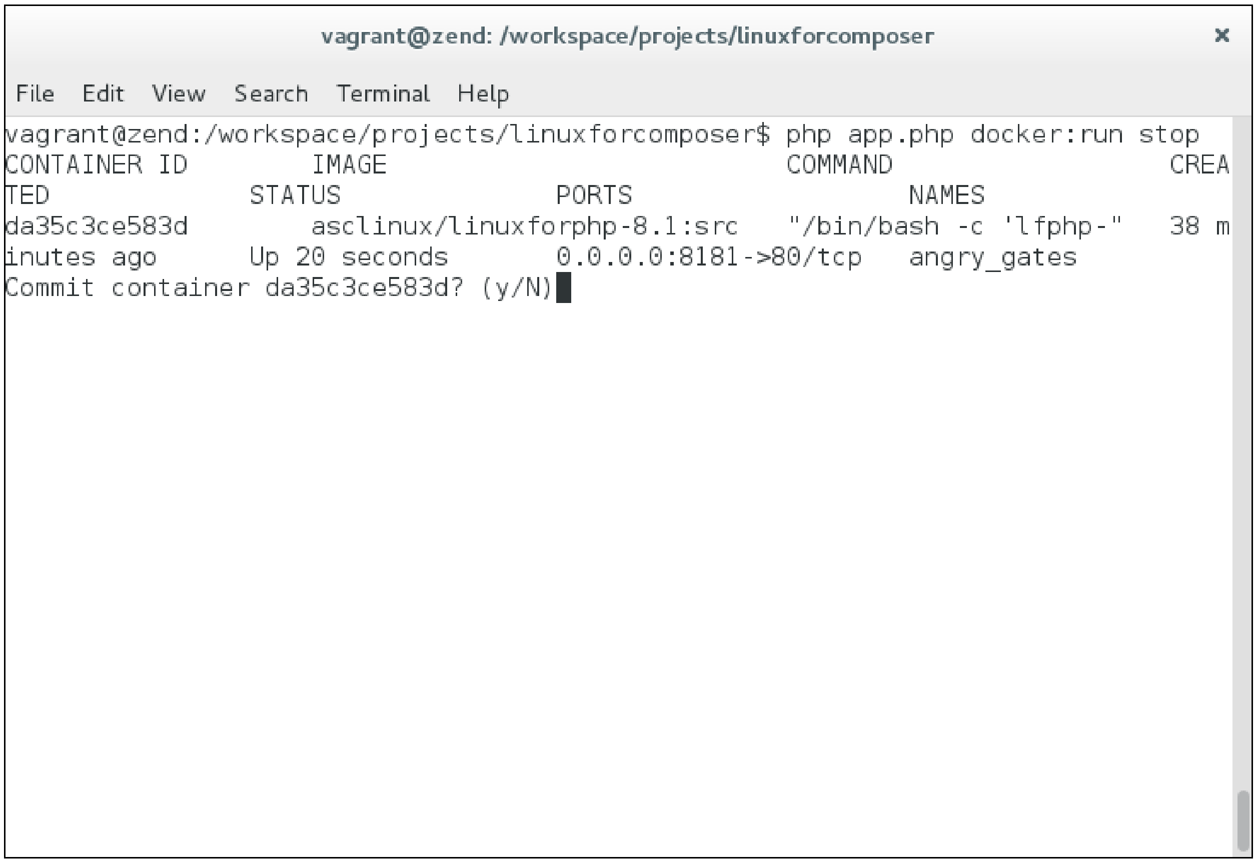
```
$ php vendor/bin/linuxforcomposer.phar docker:run start
```

1.3.2 linuxforcomposer docker:run stop

In order to stop all the containers that were started using **Linux for Composer**, please enter the following command:

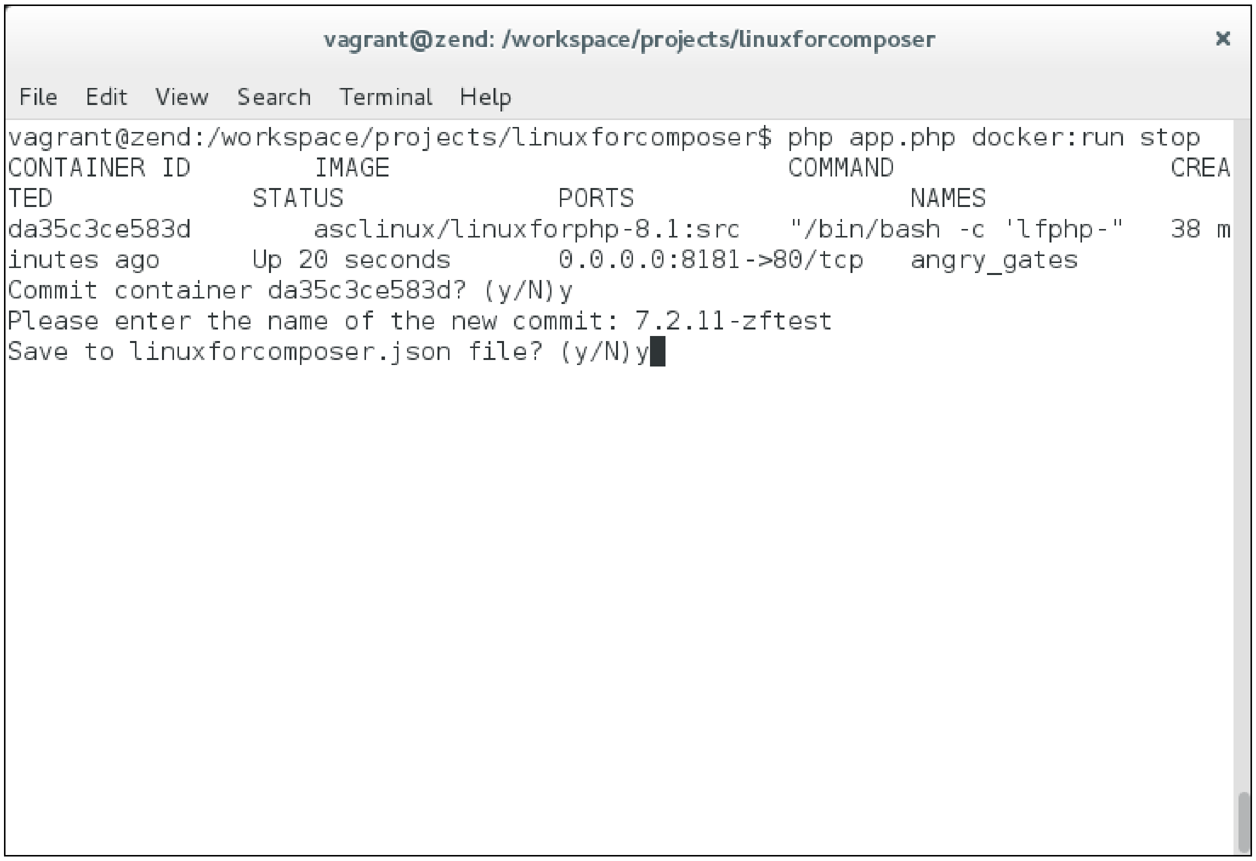
```
$ php vendor/bin/linuxforcomposer.phar docker:run stop
```

The `docker:run stop` command will automatically ask you if you want to commit each and every container that you have started before stopping and removing them.


vagrant@zend: /workspace/projects/linuxforcomposer
File Edit View Search Terminal Help

```
vagrant@zend:/workspace/projects/linuxforcomposer$ php app.php docker:run stop
CONTAINER ID        IMAGE                          COMMAND                  CREA
TED             STATUS              PORTS                  NAMES
da35c3ce583d        asclinux/linuxforphp-8.1:src   "/bin/bash -c 'lfphp-"   38 m
inutes ago      Up 20 seconds       0.0.0.0:8181->80/tcp   angry_gates
Commit container da35c3ce583d? (y/N)
```

If you do wish to save them, you will be asked to give each commit a unique name and you will also be asked if you wish to save the new name to the `linuxforcomposer.json` file for use the next time you start containers with **Linux for Composer**.


vagrant@zend: /workspace/projects/linuxforcomposer
File Edit View Search Terminal Help

```
vagrant@zend:/workspace/projects/linuxforcomposer$ php app.php docker:run stop
CONTAINER ID        IMAGE                          COMMAND                  CREA
TED             STATUS              PORTS                  NAMES
da35c3ce583d        asclinux/linuxforphp-8.1:src   "/bin/bash -c 'lfphp-"   38 m
inutes ago      Up 20 seconds       0.0.0.0:8181->80/tcp   angry_gates
Commit container da35c3ce583d? (y/N)y
Please enter the name of the new commit: 7.2.11-zftest
Save to linuxforcomposer.json file? (y/N)y
```

1.3.3 linuxforcomposer docker:run stop-force

In order to force stop all the containers that were started using **Linux for Composer** without being asked to commit each and every container, please use the following command:

```
$ php vendor/bin/linuxforcomposer.phar docker:run stop-force
```

The `docker:run stop-force` command will automatically stop and remove each and every container that you have started.

1.3.4 linuxforcomposer docker:run deploy

In order to deploy your current configuration file to the **Linux for PHP Cloud Services**, please use the following command:

```
$ php vendor/bin/linuxforcomposer.phar docker:run deploy
```

The `docker:run deploy` command will automatically post your configuration to the **Linux for PHP Cloud Services**.

Note: Please note that some configurations might be restricted due to the limitations of your service plan. Please see https://linuxforphp.com/account for more details on your service plan.

1.4 What's New

1.4.1 What's New in Version 2.0.9 (2020-07-10)

- Fixes some issues when some configuration settings are missing.
- Fixes an issue when the 'build' and 'run' keywords are used in scripts.
- Optimizes the 'Parsejson' command.

1.4.2 What's New in Version 2.0.8 (2020-06-15)

- Fixes an issue when using only a Dockerfile as the minimum configuration.
- Fixes some failures when using Linux for Composer without Composer.

1.4.3 What's New in Version 2.0.7 (2020-05-18)

- Fixes an issue with clean restarts on local/Windows computers.

1.4.4 What's New in Version 2.0.6 (2020-05-04)

- Fixes an issue when restarting an LfC container using Docker.

1.4.5 What's New in Version 2.0.5 (2020-05-01)

- Fixes an issue when reading from environment variables.

1.4.6 What's New in Version 2.0.4 (2020-04-30)

- Fixes an issue when the JSON file is invalid.
- Fixes a minor regression when compiling PHP from source.
- Replaces the flocker driver by the local driver for Docker shared volume creation.

1.4.7 What's New in Version 2.0.3 (2020-04-09)

- Adds the shared Docker volume size feature for the LfPHP Cloud.

1.4.8 What's New in Version 2.0.2 (2020-03-19)

- Fixes an issue when changing from a Dockerfile to Linux for Composer in order to start containers.
- Updates the LfPHP client to take into account the new Cloud API.
- Adds the '–version' option.

1.4.9 What's New in Version 2.0.1 (2020-03-09)

- Fixes an issue with volume paths on Windows.

1.4.10 What's New in Version 2.0.0 (2020-02-24)

- Adds new Dockerfile and docker-compose functionality.
- Adds data persistence through mounted storage.
- Adds a 'stop-force' command.
- Updates the PHP versions to the Linux for PHP 8.2.0 pre-compiled versions.
- Adds new deployment functionality for the LfPHP Cloud.

1.4.11 What's New in Version 1.0.2 (2019-01-15)

- Updates the PHP versions to the Linux for PHP 8.1.3 pre-compiled versions.

1.4.12 What's New in Version 1.0.1 (2019-01-13)

- Fixes an issue whereby the Linux for Composer PID file could be deleted by the 'composer update' command.
- Fixes an issue with the JSON formatting of the main configuration file.

1.4.13 What's New in Version 1.0.0 (2018-11-07)

- Adds a new 'commit' feature when stopping containers.
- Adds official documentation.
- Fixes an issue with Docker commands on Windows 10.

1.5 Linux for PHP License

1.5.1 Apache License

CHAPTER 2

Indices and tables

- genindex
- search

Index