Lampman - LAMP server container Orchestration

Basic usage

1. First, go to the project directory and run the following command:

   ```
   lamp init
   ```

   `(project-dir)/.lampman` directory is created for lampman settings.

2. Start the servers.

   ```
   lamp up
   ```

3. You now have a LAMP (Linux, Apache, MySQL, PHP) environment in the `(project-dir)/public_html/` directory.
   Start development by accessing http://localhost/ in a browser.

4. If you want to close the containers, use the following command:

   ```
   lamp down
   ```

That's it.
From the next time, `lamp init` is unnecessary.

Below are detailed settings and command descriptions.

Initial LAMP environment

Services/Apps	Status	External ports	Version	Memo
Linux	**Enabled**		CentOS 7.6	
Apache	**Enabled**	80, 443		`public_html/` is published
MySQL	disabled	3306	MySQL 5.6	with Xdebug
PostgreSQL	disabled	5432	PostgreSQL 9.4	
PHP	**Enabled**		PHP 5.4.16	
Perl/CGI	disabled		perl 5.16.3	

Services/Apps	Status	External ports	Version	Memo
MailDev	**Enabled**	9981	MailDev 1.1.0	
Postfix	**Enabled**		Postfix 2.10	All mail passing through Postfix is relayed to `MailDev`.

*You can easily specify PHP, MySQL, and PostgreSQL versions in the `.lampman/config.js`.
*The actual version may be different.

Docker images to use

Internally, the container is started using docker-compose.
Basically, the following image is used, but this can be changed to a self-made image etc. in the `config.js`.

Images	Description
Docker Hub: `kazaoki/lampman`（未リンク）	LAMP base image: Linux, Apache, Postfix, MailDev
Docker Hub: `kazaoki/phpenv`	This is a version-specific PHP container compiled with phpenv.
Docker Hub: `mysql`	MySQL official image
Docker Hub: `postgres`	PostgreSQL official image

Example project directory

```
(project-dir)/
    │
    │  // Version control, Task runner, etc.
    ├─ .git/
    ├─ gulp.js
    ├─ package.json
    ├─ ...
    │
    │  // Publish web root. (mount to /var/www/html)
    ├─ pulic_html/
    │   ├─ index.php
    │   └─ ...
    │
    │  // Lampman settings
    └─ .lampman/
        ├─ lampman/
        │   ├─ before-starts.sh
        │   └─ entrypoint.sh
        ├─ mysql/
        │   ├─ before-entrypoint.sh
        │   └─ main.sql
```

```
    ├─ postgresql/
    │    ├─ before-entrypoint.sh
    │    └─ sub.sql
    ├─ config.js
    ├─ docker-compose.override.yml
    ├─ docker-compose.yml
    └─ README.md
```

Internal flow of `lamp up` command

1. Auto update `.lammpman/docker-compose.yml` from `config.js`
2. `docker-compose up -d` is executed internally.
   This loads `.lammpman / docker-compose.yml` and `.lammpman / docker-compose.override.yml`.
3. If a PHP version is specified, start the corresponding PHP version container.
4. If there is a MySQL setting, start the corresponding MySQL container.
5. If there is a PostgreSQL setting, start the corresponding PostgreSQL container.
6. Finally, the Lampman base image `kazaoki/lampman` is executed and various servers are started.

Install

```
npm i lampman -g
```

That's it.

`lamp` Command

Command	Description
`lamp init`	Initialize
`lamp up`	Start servers. `lamp up -c` ... Start after forcibly deleting all other running containers. (Volume is kept) `lamp up -cv` ... Start after forcibly deleting all other running containers and volumes. (Keep locked volume)
`lamp down`	Stop and delete servers. `lamp down -v` ... Also delete related volumes. (Keep locked volume)
`lamp mysql`	MySQL operation (if no option is specified, the mysql client is executed) `-d, --dump <to>` ... Dump (Output destination can be specified) `-r, --restore` ... Restore. (Dump selection) `-c, --cli` ... Enter the console.

Command	Description
`lamp psql`	PostgreSQL operation (if no option is specified, the psql client is executed) `-d, --dump <to>` ... Dump (Output destination can be specified) `-r, --restore` ... Restore. (Dump selection) `-c, --cli` ... Enter the console.
`lamp logs`	Error log monitoring `-g, --group <name>` ... You can specify a log group name. The first one if not specified
`lamp ymlout`	Standard output as setting data as yml (relative to project root)
`lamp version`	Swho version

common options

Option	Description
`-m, --mode <mode>`	Execution mode can be specified. If not specified, the `.lampman /` directory will be referenced. For example, if `-m product` is specified, the`.lampman-product /`directory will be created.
`-h, --help`	Each help is displayed. ex: `lamp -h lamp mysql -h`

For production

You can create a `docker-compose.yml` for production in the project root with the following command:

```
lamp product-yml
```

This is the same as the following command.

```
lamp ymlout -m product > (project-dir)/docker-compose.yml
```

This command is registered as an extra command.

Extra commands

You can register additional commands with config.js. You can choose to run on the host side or on the container side.

Description of `config.js`

```
const __TRUE_ON_DEFAULT__ = 'default'===process.env.LAMPMAN_MODE;

/**
 * load modules
 */

 /**
 * export configs
 */
module.exports.config = {

    // Lampman
    lampman: {
        project: 'lampman-test',
        image: 'kazaoki/lampman',

...
```