Docker环境配置（win10配置）

本文介绍在Docker容器中，快速部署win环境的完整步骤。

最终完成的环境我们暂且叫：**DNMP**（即 Docker + Nginx + MySQL + PHP）。

最终实现一键部署WNMP环境的目的，特点：

1. **完全开源**
2. 支持**多版本PHP切换（PHP5.4、PHP5.6、PHP7.2...)**
3. 支持绑定**任意多个域名**
4. PHP**源代码位于host**中
5. **MySQL data位于host中**
6. 所有**配置文件可在host中**直接修改
7. 所有**日志文件可在host中**直接查看
8. 内置完整**PHP扩展**安装命令
9. 实现**一次配置，可在任何支持Docker系统使用**

1 快速使用

github**仓库地址**：**https://github.com/xiaohui007/dnmp.git**

1、安装git、docker和docker-compose。

2、使用git下载完整代码：

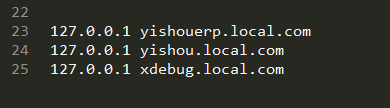
$ git clone https://github.com/xiaohui007/dnmp.git

3、用docker-compose命令启动容器，首次使用需要下载镜像，会稍慢：

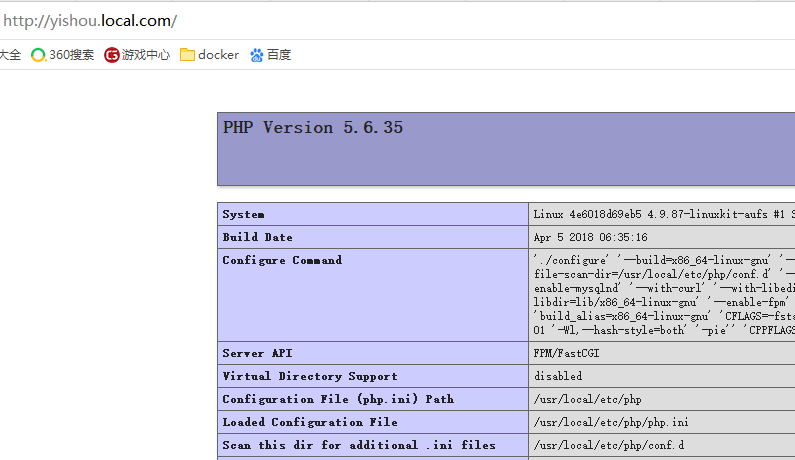
$ cd dnmp

$ docker-compose up

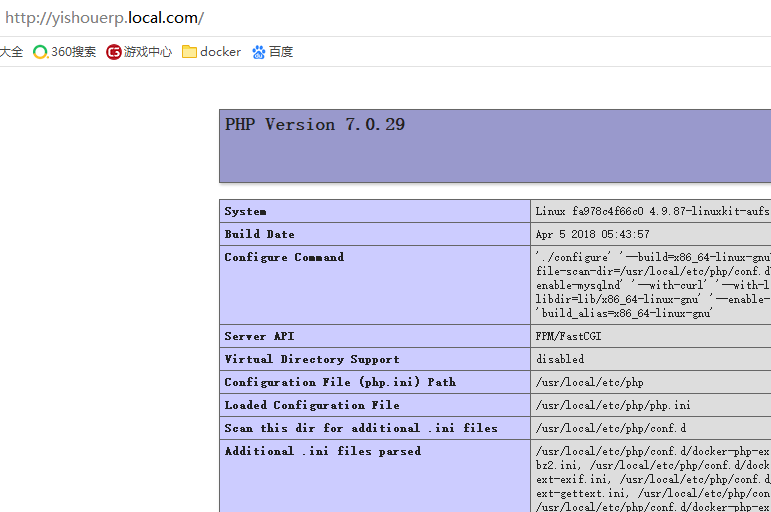
4、修改host（C:\Windows\System32\drivers\etc）文件



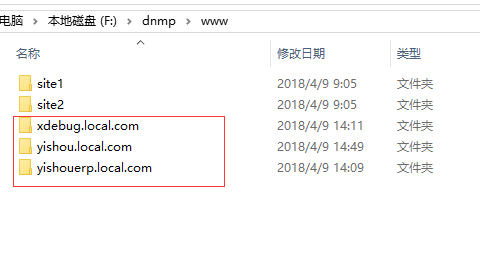
4、然后在浏览器中访问对应的网址，就可以看到如下页面：







5、源代码在：./www/ 对应的目录下。



2 安装docker和docker-compose 本教程适合window系统

docker下载：

[Docker for Windows](https://store.docker.com/editions/community/docker-ce-desktop-windows) 在Windows上运行Docker。系统要求，Windows10x64位，支持Hyper-V。（安装文件可以到文档附带的安装文件夹查找）

如果您的电脑版本过旧，可以使用 [Docker Toolbox](https://www.docker.com/toolbox) 在Windows或者Mac上运行Docker。适用于Mac OS X 10.8+ 或者 Windows 7/8.1。

具体安装步骤：

<https://blog.csdn.net/shi1451042748/article/details/52996046>

备注：win10 安装过程中的常见报错

1. Windows 10安装Docker 报“Hardware assisted virtualization and data execution protection must be enabled”

解决方案：<https://my.oschina.net/u/2289161/blog/1647017>

1. <https://segmentfault.com/a/1190000006799421>

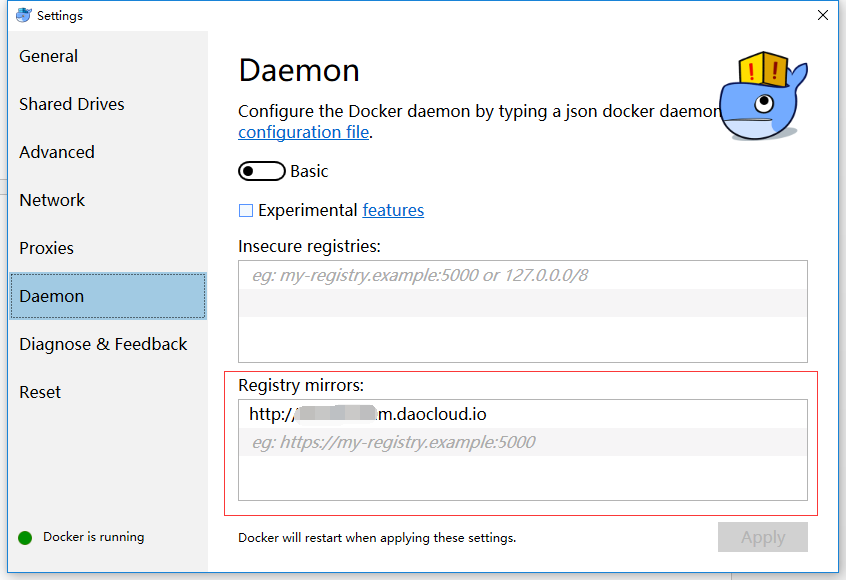
3 使用国内镜像仓库

安装步骤已经有相关说明，这边提供一下国内的加速器。

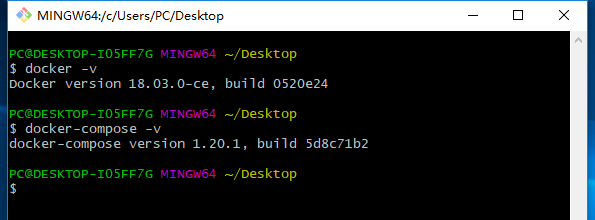
到[DaoCloud](https://www.daocloud.io/" \t "_blank) 注册，并使用加速器，链接如下：

[https://www.daocloud.io/mirror#accelerator-doc](https://www.daocloud.io/mirror%23accelerator-doc)



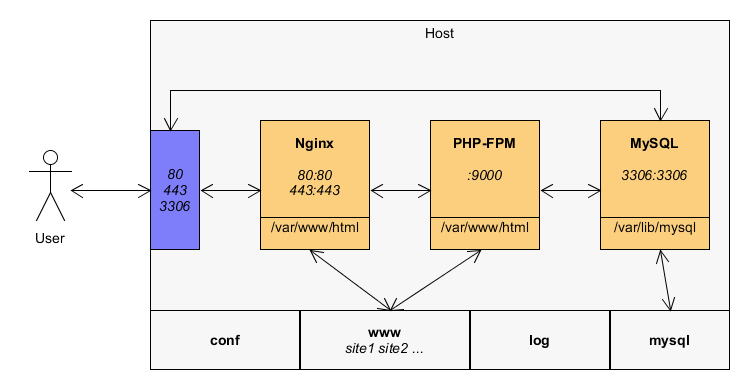


Docker，docker-compose安装完成测试：



4 目录说明

大致框架如下：



4.1 目录结构

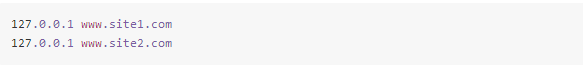
目录结构如下：



4.2 站点部署

本文有默认加了两个站点：www.site1.com（同localhost）和www.site2.com。

要在本地访问这两个域名，需要修改你的hosts文件，添加以下两行：



其中，www.site2.com为**支持SSL/https和HTTP/2的示例站点**。

因为站点2的SSL采用**自签名方式**，所以浏览器有安全提示，继续访问就可以了，自己的站点用第三方SSL认证证书替换即可。

如果只用到站点1，把站点2相关的目录和配置文件删除：

./conf/nginx/conf.d/certs/site2/

./conf/nginx/conf.d/site2.conf

./www/site2/

重启容器内的Nginx生效：

docker exec -it dlnmp\_nginx\_1 nginx -s reload

4.3 HTTPS使用

**在容器中我们也是可以用HTTPS的**，具体的配置请参考如下文件：

./conf/nginx/conf.d/site2.conf

如果是自签名，可以用**[廖雪峰](http://www.liaoxuefeng.com/article/0014189023237367e8d42829de24b6eaf893ca47df4fb5e000)**提供的一个自动生成认证文件、私钥脚本：gencert.sh，

这个脚本已经放在项目中，在这个目录下：

./conf/nginx/conf.d/certs/site2/

在Bash中输入：

$ ./gencert.sh

输入一次域名，和几次密码（内容随意）后，就会生成几个认证文件。

其中自签名情况不需要.csr和.origin.key后缀的文件。

然后修改Nginx配置文件，配置SSL支持就可以了。

5 docker-compose.yml文件

如下是docker容器的运行配置docker-compose.yml的内容：



这里我们用了**nginx的alpine镜像**，以及php-fpm和MySQL的最新镜像。

这里默认加了redis容器，不需要的话可以把文件的**最后**4**行**以及php-fpm块下的- redis:redis删除。

5.1 为什么用nginx:alpine？

因为相比nginx:latest，nginx:alpine有几点优势：

1. 用的是最新版nginx镜像，功能与nginx:latest一模一样
2. alpine镜像用的是Alpine Linux内核，比ubuntu内核要小很多。
3. nginx:alpine默认支持http2。

Alpine的更多请看《[Alpine Linux，一个只有5M的Docker镜像](http://www.infoq.com/cn/news/2016/01/Alpine-Linux-5M-Docker)》。

如要进入alpine容器，命令是（后面的路径不是/bin/bash）：

$ docker exec -it dnmp\_nginx\_1 /bin/sh

5.2 站点根目录挂载

为什么站点根目录在Nginx和PHP-FPM都这样挂载？

./www/:/var/www/html/

我们知道，Nginx配置都有这样一项：

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

其中，$document\_root就是server块下root所指的路径：

server { #...     root /var/www/html; #... }

这里$document\_root就是/var/www/html。

如果Nginx和PHP-FPM在同一主机，Nginx会通过9000端口把这个**目录值**和**脚本URI**传给PHP-FPM。

PHP-FPM再通过9000端口接收Nginx发过来的目录值和脚本URI，发给PHP解析。

PHP收到后，就到指定的目录下查找PHP文件并解析，完成后再通过9000端口返回给Nginx。

**如果Nginx和PHP-FPM在同一个主机里面，PHP就总能找到Nginx指定的目录。**

但是，如果他们在不同的容器呢？

未做任何处理的情况，Nginx容器中的站点根目录，PHP-FPM容器肯定不存在。

所以，这里需要**保证Nginx和PHP-FPM都挂载了Host的**./www/**，并且都挂载在容器的：**/var/www/html。

（当然，你也可以指定别的目录，确保统一即可）

5.3 修改docker-compose.yml文件之后？

如果容器已经生成，回头再编辑docker-compose.yml，用

docker-compose up

命令会直接启动原来的容器，**修改的内容不会体现在启动的容器里**。

所以，要使修改的docker-compose.yml生效，需要以下4步：

$ docker stop dnmp\_nginx\_1 # 第一步：停止容器 $ docker rm dnmp\_nginx\_1 # 第二步：删除容器 # !!第三步：重启Docker服务!! $ docker-compose up -d --no-deps --build mysql # 第四步：重新启动容器

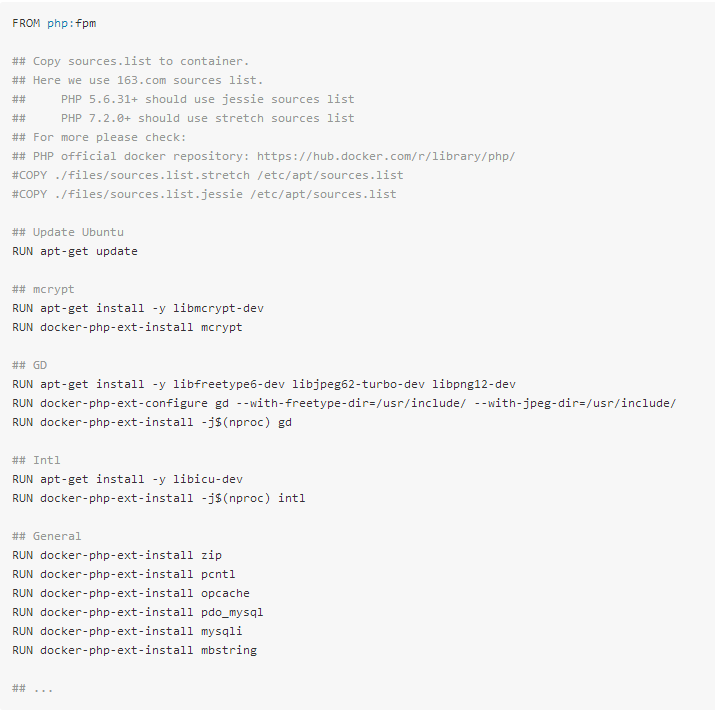
其中最后一条命令参数作用：

* -d：后台执行
* --no-deps：不启动link的容器
* --build：启动容器前先构建镜像

6 Dockerfile文件

因为PHP-FPM构建稍微复杂，涉及到很多扩展。

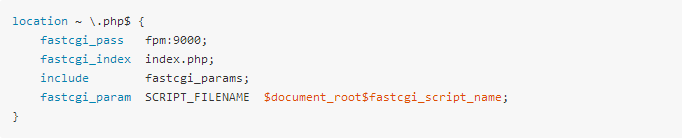
所以单独用Dockerfile文件构建PHP-FPM：



**完整Dockerfile请移步本项目GitHub仓库。**

7 nginx配置文件

这里，我们还需要稍微修改nginx配置文件：



这里，我们仅需把fastcgi\_pass改成：fpm:9000。

也就是，PHP-FPM的侦听主机改成：Nginx links PHP-FPM容器的**别名**，在docker-compose.yml文件里面我们设置为fpm。

**这样Nginx容器启动的时候，就会自动修改自己的/etc/hosts，让fpm指向php-fpm容器的IP。**

修改之后，重启容器中的nginx：

$ docker exec dlnmp\_nginx\_1 nginx -s reload

其中，dlnmp\_nginx\_1是容器的名称，也可以指定容器的ID。

8 站点根目录写权限

有时候，我们的PHP代码需要某个目录的写权限。

比如，Wordpress的wp-content/uploads目录，只有写权限我们才能正常上传文件。

默认的，容器中的/var/www/html目录属于root，我们需要修改为www-data，PHP才能正常写目录。

先进入到容器中：

$ docker exec -it dnmp\_php-fpm\_1 /bin/bash

然后修改目录权限：

$ chown -R www-data:www-data /var/www/html

这样PHP就能正常读写目录了。

9 数据库

9.1 使用MySQL

在docker-compose.yml文件中，我们指定了MySQL数据库root用户的密码为123456。

所以，我们就可以在主机中通过：

$ mysql -h 127.0.0.1 -u root -p

输入密码，就可以进入MySQL命令行。

**说明：这里MySQL的连接主机不能用**localhost**，因为MySQL客户端默认使用unix socket方式连接，应该直接用本地IP。**

在PHP代码中的使用方式与在主机中使用稍有不同，如下：

$pdo = new PDO('mysql:host=mysql;dbname=site1', 'root', '123456');

其中，host的值就是在docker-compose.yml里面指定的MySQL容器的名称。

这是因为PHP代码是在FPM容器中，FPM容器启动时会自动在/etc/hosts中加上：

172.17.0.2 mysql 11e55f91c4c3 dlnmp\_mysql\_1

就是说，mysql自动指向了MySQL容器动态生成的IP。

9.2 使用Redis

Redis使用和MySQL类似。

在主机和容器内部都通过地址127.0.0.1，端口6379访问。

PHP则是跨容器访问，host参数用redis（links指定的名称），端口用6379。