^2\_3^2workroom3工作室

--

//// 使用 Docker / Docker Compose 部署 Swoft 应用

https://segmentfault.com/a/1190000017297770?utm\_source=tag-newest

--

--

1、Swoft

首个基于 Swoole 原生协程的新时代 PHP 高性能协程全栈框架，内置协程网络服务器及常用的协程客户端，常驻内存，不依赖传统的 PHP-FPM，全异步非阻塞 IO 实现，以类似于同步客户端的写法实现异步客户端的使用，没有复杂的异步回调，没有繁琐的 yield, 有类似 Go 语言的协程、灵活的注解、强大的全局依赖注入容器、完善的服务治理、灵活强大的 AOP、标准的 PSR 规范实现等等，可以用于构建高性能的Web系统、API、中间件、基础服务等等。

--

--

2、Swoft的Docker镜像

使用 docker 安装 swoft 其实听起来比较怪怪的，swoft 是一套 php 框架，依赖 swoole 扩展，说 docker 安装 swoft，其实是 docker 安装 swoft 运行所需的组件依赖和环境。

swoft 框架运行环境所需的依赖和环境挨个安装搭建还是需要一些时间的，比如 php 版本 >= 7.1，swoole 版本 >= 2.1，而且还要安装 hiredis 来协助开启 swoole 的异步 redis 客户端，同时要求 swoole 开启协程模式等。

所以呢，为了节省我们的时间，官方提供了一个 docker 镜像包，里面包含了 swoft 运行环境所需要的各项组件：php 7.1+、swoole 2.1+ --enable-async-redis-client --enable-coroutine、composer、pecl。

我们只需要下载镜像并新建一个容器，这个容器就提供了 swoft 框架所需的所有依赖和环境，将宿主机上的 sowft 项目挂载到 swoft 镜像的工作目录 /var/www/swoft 下，就可以继续我们的开发或生产工作了。让你从 swoft 略繁琐的依赖和环境搭建中解放出来，直接进入业务开发工作中去。

一开始我没理解好这个 swoft 镜像，镜像里自带的框架其实是单纯的用来体验的，我一直误以为要编辑镜像的 swoft 框架源码做开发。

需要特别注意的是，sowft 镜像的 entrypoint 命令（运行初始化命令）是 ENTRYPOINT ["php", "/var/www/swoft/bin/swoft", "start"] 。即 swoft 服务会随着容器而启动，这就要求我们如果选择将宿主机上开发用的 swoft 项目挂载到容器工作目录时，需已完全安装才可以（使用 composer 安装好各依赖）。同时容器使用 swoft服务 作为前置进程，若我们想停止/启动来重新载入 swoft服务 时，容器也会跟随退出，这样就略有不便了。所以，为了后续开发方便，我们应分离 swoft 服务作为容器的前台进程，使得在容器内 重启/停止 swoft 服务不影响容器自身运行。当然，如果只是单纯的体验 swoft，直接创建并启动容器即可，镜像中已有一套完全安装的 swoft框架。

在后面我们将给出一个只需要在宿主机上安装运维所需的 docker/docker-compose/git 即可完全借助 swoft镜像 去部署开发或生产环境的方法（修改镜像 entrypoint 到 bash 模式，然后进入镜像后使用 composer 安装依赖，启动 swoft，充分利用镜像资源）。

--

--

3、Docker 部署 swoft

--

宿主机仍需安装基本的 php / composer（或者你把自己本地开发的项目cp过来，但这样可能会导致部分组件版本不一致，还是提交业务代码 + composer.json + composer.lock 文件，排除 vendor 目录，在线上服务器再 composer install 一遍最为规范）。

--

(1)、在宿主机创建 swoft 项目（宿主机需实安装基础的 php 环境来使用 composer）

//// 创建项目

composer create-project --prefer-dist swoft/swoft swoft [--dev] && cd swoft

// 或者

//// 克隆项目

git clone git@github.com:swoft-cloud/swoft.git && cd swoft && composer install && cd swoft

--

(2)、拉取 swoft 镜像 创建 swoft 容器 并将宿主机上安装好的 swoft 项目挂载到 swoft 容器的工作目录

// 拉取 swoft 镜像

// 关联本地 swoft 项目目录到镜像的项目目录(/var/www/swoft)

// 映射主机 8081 端口 到 容器 80 端口

// 容器命名为 mySwoft

// 守护模式启动

docker run -v $(pwd):/var/www/swoft -p 8081:80 --name mySwoft -d swoft/swoft

// 查看容器是否运行

docker ps

// 查看容器日志

docker logs mySwoft

--

(3)、进入 swoft 容器 shell

// 交互模式执行 mySwoft 容器的 bash

docker exec -it mySwoft bash

//

// stop 会停止容器所以会退出 shell 后用 docker start mySwoft 启动就好

root@cce12db9add3:/var/www/swoft# php bin/swoft start|stop|reload

// 因我们将宿主机上的swoft项目挂载到了swoft容器的项目目录/var/www/swoft 所以后期开发修改宿主机上的项目即可(可以使用PS的FTP同步工具)

//

// 可以在 swoft 的容器 shell 里通过命令查看相应的组件版本

root@cce12db9add3:/var/www/swoft# php -v

root@cce12db9add3:/var/www/swoft# php --ri swoole

root@cce12db9add3:/var/www/swoft# composer -V

root@cce12db9add3:/var/www/swoft# pecl -V

--

--

4、Docker Composer 部署 Swoft

宿主机仍需安装基本的 php / composer（或者你把自己本地开发的项目cp过来，但这样可能会导致部分组件版本不一致，还是提交业务代码 + composer.json + composer.lock 文件，排除 vendor 目录，在线上服务器再 composer install 一遍最为规范）。swoft 项目中是有 docker-compose.yml 文件的。

//

//// docker-compose.yml

version: '3'

services:

swoft:

image: swoft/swoft:latest

# build: ./

ports:

- "80:80" #端口映射(宿主端口映射到容器端口)

volumes:

- ./:/var/www/swoft #挂载当前路径下的本地swoft项目到镜像项目路径

stdin\_open: true #打开标准输出

tty: true #打开 tty 会话

privileged: true #给与权限，比如创建文件夹之类的

entrypoint: ["php", "/var/www/swoft/bin/swoft", "start"] #入口启动命令，即启动 swoft 服务

//

使用方法自然比直接用 docker 方便些，不过依旧是要在宿主机上先创建一个 swoft 项目。

(1)、1、在宿主机创建 swoft 项目（宿主机需实安装基础的 php 环境来使用 composer）

//// 创建项目

composer create-project --prefer-dist swoft/swoft swoft [--dev] && cd swoft

// 或者

//// 克隆项目

git clone git@github.com:swoft-cloud/swoft.git && cd swoft && composer install && cd swoft

//

(2)、使用 docker-compose 来编排启动容器

//// 编辑 docker-compose.yaml 文件 给容器自定义个名字

version: '3'

services:

swoft:

image: swoft/swoft:latest

container\_name: mySwoft #给容器自定义个名称便于管理

# build: ./

ports:

- "80:80" #端口映射

volumes:

- ./:/var/www/swoft #挂载当前路径下的本地swoft项目到镜像项目路径

stdin\_open: true #打开标准输出

tty: true #打开 tty 会话

privileged: true #给与权限，比如创建文件夹之类的

entrypoint: ["php", "/var/www/swoft/bin/swoft", "start"] #入口启动命令，即启动 swoft 服务

//

// 启动容器

docker-compose up -d swoft

// 查看容器是否成功运行

docker ps

// 进入容器shell

docker exec -it mySwoft bash

--

--

5、在未安装 PHP 环境的宿主机上部署 swoft

前面两种部署 swoft 的方法都需要在宿主机上安装 php 基础环境来使用 composer 安装好本地 swoft 项目的依赖组件，才能与 swoft 镜像的工作目录挂载，启动容器（因为容器的入口命令就是直接启动 swoft，如果我们挂载本地未安装好依赖的 swoft 项目到镜像工作目录，那容器就会启动失败退出了），下面我们介绍一种不需要在宿主机上安装 php / composer 的方法。

//

(1)、拉取 swoft（拉取就好，不需要安装依赖）

git clone git@github.com:swoft-cloud/swoft.git && cd swoft

//

(2)、直接使用 docker 镜像

# -it 开启标准输入及终端

# --entrypoint 覆盖镜像内默认启动 swoft 服务的设定

# -d 守护模式

# 这样便使得容器在启动时会创建一个 bash 作为前置进程 而不启动 swoft 服务

docker run -it -d -p 80:80 \

-v $(pwd):/var/www/swoft \

--name mySwoft \

--entrypoint="" \

swoft/swoft bash

# 启动后进入容器

docker exec -it mySwoft bash

# 使用容器内的 composer 安装依赖 此时的工作目录已于宿主机的swoft项目关联

compose intall [--no-dev]

# 启动 swoft

php bin/swoft start

# 此时停止 swoft 也不会导致容器退出

php bin/swoft stop

//

(3)、使用 docker-compose

//// 编辑 docker-compose.yml 文件，开启 stdin\_open（等同于docker的 -i）, tty（等同于docker的 -t）, entrypoint 改为 bash。

version: '3'

services:

swoft:

container\_name: mySwoft

image: swoft/swoft:latest

# build: ./

ports:

- "8082:80" #映射宿主机8082端口到容器80端口

volumes:

- ./:/var/www/swoft # 将宿主机的当前项目目录挂载到容器的工作目录"/var/www/swoft"

stdin\_open: true #一定要开启此项否则容器会因 bash 执行完退出

tty: true # 开启会话终端

privileged: true

# entrypoint: ["php", "/var/www/swoft/bin/swoft", "start"]

entrypoint: ["bash"] # 改为此命令后 启动容器时默认不会启动 swoft 所以即使框架依赖未安装 也不会影响容器启动

//

// 保存 docker-compose.yml 后启动容器

docker-compose up -d swoft

//

// 进入容器 shell 使用容器中的 composer 安装框架依赖

# 进入容器shell

docker exec -it mySwoft bash

# 会默认在 swoft 镜像的工作目录 /var/www/swoft 此目录以和宿主机的swoft项目目录映射在一起了

# 用容器内的 composer 安装依赖

composer install [--no-dev]

# 启动 swoft

php bin/swoft start|stop|restart

//

这样使得宿主机完全省去了还要事先简单安装下 php / composer 的工作，完全利用镜像提供的现成的环境~。

--

--

--