链滴

Solo 博客系统基于 Docker 的搭建

作者：ReyRen

原文链接：https://ld246.com/article/1563623196459

来源网站：链滴

环境

我自己购买了云主机，至于哪里有卖就不需要强调了吧. 顺手趁打折我就冲了个域名: reyren.cn. 如果好好的去经营自己的点点滴滴, 这些几十一百的不要太在意, emmm扯远了. 在我配置好服务器基础环添加域名解析后(Ubuntu 16.04 LTS)， 我就折腾的部署了Solo. 索性就在几分钟后还是检测出来域名用于了网站等性质的活动, 直接封掉要求备案了. 在经历一番流程备案后，审核耗时10个工作日, 终于见天日了.

在官方的README中也不推荐用war发布包和源码的方式进行安装, 因为这样不利于bug的及时修复和统的更新. 所以这里也懒得折腾，直接用最推荐的也是最方便的docker形式进行部署.

所需环境:

- docker
- docker-compose
- solo-in-docker

Docker

Docker是一个通过容器进行应用程序的部署的开源技术. 虽然岁数不大, 但是在主流的linux发行版上是被装烂的. Docker主要的创新点是可以"打包"一个应用程序或者是一系列的服务在容器中. 一个Docker容器其实就是包含了所有依赖库和组件的应用例程. 从时间工程的角度看(粗力度), 一个容器就像是减了操作系统关联性的虚拟机. 一个Docker容器产生于image, image产生于打包的应用. image可以包完整的操作系统或者预装的应用. 官方Docker分为CE(Community Edition)和EE(Enterprise Edition). 我们这里就针对于CE for Ubuntu的安装. (如果你想要直接"Ubuntu的默认源安装，比较陈旧吧，我般都会添加官方源，便于安装也便于日后更新. 如果没有的话，推荐使用源码安装")

```
# Uninstall old versions
$sudo apt-get remove docker docker-engine docker.io
# Update the apt package index
```

```
$sudo apt-get update
# Install packages to allow apt to use a repository over HTTPS
$sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
# Add Docker's official GPG key
$curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
# Verify that you now have the key with the fingerprint 9DC8 5822 9FC7 DD38 854A E2DB 8D
1 803C 0EBF CD88, by searching by last 8 characters
$sudo apt-key fingerprint 0EBFCD88
# Use the following command to set up the stable repository. You always need the stable one,

#even if you want to install builds from the edge or test repo as well. To add the edge or test
epo, add the word edge or test after the word stable (stable_release -cs获取到ubuntu版本)
$sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(l
b_release -cs) stable"
#Update the apt package index
$sudo apt-get update
# Install the latest version of Docker CE, or go to the next step to install a specific version.
$sudo apt-get install docker-ce
# List the available versions
$apt-cache madison docker-ce
$sudo apt-get install docker-ce=<VERSION>
# The docker daemon already running automatically
```

这样的话要想升级docker ce只需要sudo apt-get update更新了源后找到最新的版本直接安装了就覆盖之前的版本.

Docker-compose

这个就是如雷贯耳的容器编排工具. 编排工具就是定义和运行多个容器应用的工具. YAML文件就是用配置你的服务的. 然后一个命令就能启动所有的服务和应用.

```
# Download the current stable release of Docker Compose, to install a different version of C
mpose, subtitute 1.24.1 with the version of Compose you want to use.
$sudo curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compo
e-$(uname -a)-$(uname -m)" -o /usr/local/bin/docker-compose
# Apply executable permissions to the binary
$sudo chmod +x /usr/local/bin/docker-compose
# Upgrading
# If you're upgrading from Compose 1.2 or earlier, remove or migrate your existing containers
after Compose.
# This is because, as of version 1.3, Compose uses Docker labels to keep track of containers, a
d your containers need to be recreated to add the labels.
# 但是也是可以进行迁移的，因为确实存在一切情况，启动的容器中有很多数据或者服务不可中断的
行着. 但唯一的局限就是必须得用Compose 1.5.X进行标签的迁移
$ docker-compose migrate-to-labels
# Alternatively, if you're not worried about keeping the, you can remove them. Compose just
reates new one.
$ docker container rm -f -v myapp_web_1 myapp_web_2....
# Uninstallation with curl installed
$sudo rm /usr/local/bin/docker-compose
```

solo-in-docker

摘抄自solo-in-docker, 这个工具是一条命令在docker中启动solo, 所有的配置全部都用docker-compose编排解决. 支持HTTP/HTTPS, 同样的也支持HTTP向HTTPS的升级. 在这里, 我就先以HTTP进行部然后升级到HTTPS.

```
# clone repo
$git clone https://github.com/liumapp/solo-in-docker
# put it into the root directory and change the name as solo. Following operation would use t
e root permission or add your non-root roles into the root group.
$sudo mv solo-in docker /root/solo
```

我不清楚现阶段的solo-in-docker有没有针对于之前的 issue7和 issue8进行进一步探讨及讨论, 所以就以我现阶段的问题进行搭建.

先简单介绍一下solo-in-docker的内部目录结构及功能:

● https: 这个目录就是进行证书下载让网站可以走https协议,但核心内容同http一样，所以直接带入http目录.

● http: 在这个目录中其实重要的文件就是docker-compose.yml, 这个文件就是进行容器编排的配置件，其中就是创建mysql数据库的用户名密码, 进行JDBC连接等. 至于作者原仓库中的mysql目录，现段我是不建议直接用. 这个目录就是数据库的映射文件. 也就是说每次docker-compose启动的时候, 会自动拉取最新的solo镜像, 虽然这样更新非常简单, 但是在数据备份问题上, docker容器死亡的时候, 容器内的数据会自动清除, 除非我们使用volumes构建映射关系, 这里只将最重要的mysql数据库文件射到mysql/data下. 所以说如果你直接用的话，mysql.user表中的root密码已经确定了，至于可以更，但issue8中我向作者提了一下相关的问题. 最干净的做法就是:

```
# 删除solo中的mysql
$ sudo rm mysql
# 配置docker-complse.yml文件

version: "2"

services:
  mysql:
    container_name: mysql
    image: mysql:5.5.60
    restart: always
    volumes:
      - ./mysql/data:/var/lib/mysql //mysql的数据文件存放地址
      - ./mysql/conf/mysqld.conf:/etc/mysql/mysql.conf.d/mysqld.cnf //mysql的配置文件存放地

    ports:
      - "6603:3306" //6603代表宿主机端口，3306代编容器的端口
    environment:
      - MYSQL_ROOT_PASSWORD=123456 //mysql的root账号密码
  solo:
    container_name: solo
    image: b3log/solo //直接使用最新版本的solo镜像
    restart: always
    ports:
      - "8080:8080" //如果要部署到线上的话，请改为"80:80"，同时修改下面的--listen_port=80
    environment: //此处配置solo跟mysql的连接设置，来源为上面启动的mysql容器，如果要用自己
mysql服务，那么请将mysql的servcie去掉
      RUNTIME_DB: "MYSQL"
      JDBC_USERNAME: "root"
```

```
      JDBC_PASSWORD: "123456"
      JDBC_DRIVER: "com.mysql.jdbc.Driver"
      JDBC_URL: "jdbc:mysql://mysql:3306/solo?useUnicode=yes&characterEncoding=UTF-8&u
eSSL=false&serverTimezone=UTC" //此处，因为solo跟mysql同为docker容器，所以可以直接使
容器名 + 容器端口来访问
    command: --listen_port=8080 --server_scheme=http --server_host=www.reyren.cn //按照so
o官方要求，在solo启动之初，配置solo的域名、端口，如果是本地测试的话，将host改为localhost
可
```

注意这里的mysql的密码最好改一下.

做完这些，就可以试着启动了:

```
# 在http目录下, 也就是说和.yml文件同级目录下.
$ docker-compose up -d
```

如果出现80端口被占用的报错的话, 找到nginx配置文件把端口号改一下, 或者比如我就直接先systemct stop nginx; systemctl disable nginx.

OK，再启动后发现很漂亮的done. 但是你会发现当你访问的时候确实找不到页面或者访问不到等错误. 查看日志docker logs solo, 发现找不到表的问题. 这个问题出现的原因是: 在mysql中没有发现solo这数据库，以至于无法向数据库中建立表. 本来solo-in-docker作者的仓库中的mysql/data/solo就是属已经建立好的表项, 但是之前为了干净安装, 直接干掉了mysql目录，所以这里需要我们手动的创建sol数据库.

```
# 查看当前容器运行情况, 会发现有mysql和solo
$ docker ps
# 进入容器并且打开bash
$ docker exec -it mysql bash
# 进入mysql (用之前.yml设置的密码)
$ mysql -uroot -p XXX
mysql> CREATE DATABASE `solo` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
mysql> CREATE USER 'solouser'@'%' identified BY 'solo@pwd';
mysql> GRANT ALL ON solo.* TO 'solouser'@'%';
mysql> FLUSH PRIVILEGES;
$ docker-compose down
$ docker-compose up -d
```

这样就成功了，找到管理界面进入后台进行配置吧. 在其中有个配置项每隔24小时进行一次全量备份自己的github仓库，建议打开，详情参考博文定时同步 GitHub 仓库.

到这里搭建工作就完成了.

但是呢有些人会想去搞这个mysql的东西，总是觉得不看看里面，不放心. ok, 用命令行形式是可以，是列出来的表项格式着实让人恶心. 于是我用DataGrip进行了图形话界面操作服务器中起的mysql容器. 其实很简单, 只需要注意使用链接时候的端口是6603而不是mysql默认的3306哦.

HTTP到HTTPS的迁移

在这里推荐两个我参考过的文章然后简单说一下坑和注意事项就行了, 具体重复的步骤就不再啰嗦:

首先关于证书可以参考# 从零开始安装solo博客, 但是我的建议是看到证书下载到本地并且上传到服器就ok了, 因为之前我写的一直是容器编排进行整体化管理solo博客系统, 上面文章中是单独配置启动ginx的. 问题不大, 可以参考看看是怎么映射和原理.

至于证书有了之后怎么去做, 继续参考solo-in-docker以及https数据库问题, 这离需要参考的就是直从开始已经有了证书开始.

End

大功告成, 愉快的开始积累吧+1