链滴

使用 docker-compose 一键启动 bolo 博客

作者：expoli
原文链接：https://ld246.com/article/1584849381795
来源网站：链滴

使用 docker-compose 一键启动 bolo 博客

本项目专注于使用 docker-compsoe 进行容器的编排，实现 bolo 博客的一键启动，以避免广大人群众在进行 bolo 部署时会走不必要的弯路；降低了使用门槛，同时也大大增加了维护与迁移的便利。

注意：本项目使用 nginx 的反向代理作为 bolo 的 web 服务器、目前支持一键式的http部署（默认用了80端口）、如若需要启用https访问支持，请自行进行配置。

快速开始

服务器部署

默认 bolo 的访问域名为 localhost，如果您想直接在本地在本地试用，并通过 localhost 进行访问、么无需修改任何文件、直接参考 本地快速部署测试，即可。

在进行服务器部署时，请根据需要修改 docker-compose.yaml 中的 bolo 服务中的 command 项。改完成后根据 本地快速部署测试，进行后续步骤即可。

```
  bolo:
    image: tangcuyu/bolo-solo:latest
    restart: always
    container_name: "bolo"
...
...
    networks:
      - bolo-net
    command: --listen_port=8080 --server_scheme=http --server_host=(修改为你博客的域名或ip
  --server_port=
```

启动参数说明：

- --listen_port：进程监听端口
- --server_scheme：最终访问协议，如果反代服务启用了 HTTPS 这里也需要改为 https
- --server_host：最终访问域名或公网 IP，不要带端口
- --server_port：最终访问端口，使用浏览器默认的 80 或者 443 的话值留空即可

详情请参考：Solo 用户指南

本地快速部署测试

如果你只想体验一下那么可以根据下面的命令提示进行 bolo 的快速部署。

1. **克隆本项目至本地**

```
git clone https://github.com/expoli/start-bolo-with-docker-compose.git
```

1. **进入至项目跟路径**

```
cd start-bolo-with-docker-compose
```

3. **使用 docker-compose 启动 bolo**

```
# 后台启动
docker-compose up -d

# 前台方式启动可以看到日志输出、方便进行排错
docker-compose up
```

4. **更新容器**

```
docker-compose pull && docker-compose up -d
```

5. **删除容器与docker网络（但保留mysql数据库）**

```
docker-compose down
```

6. **完全删除**

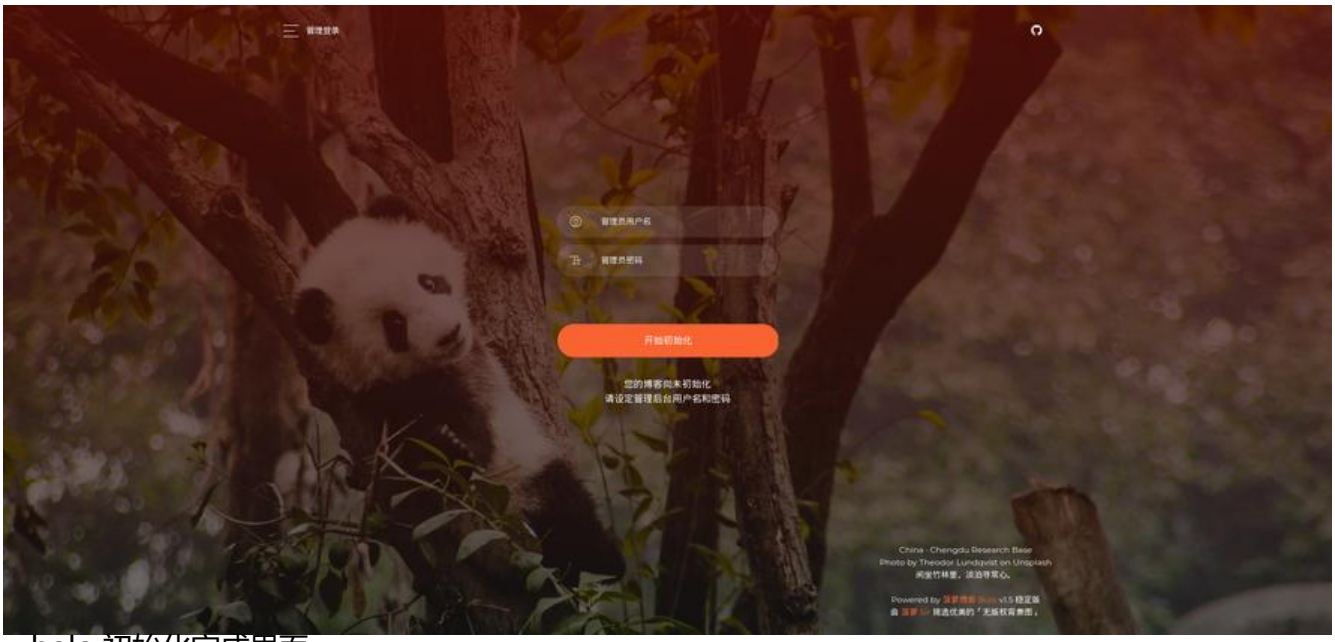
如果你想完全卸载 bolo 只需要删除本项目文件夹即可、**因为mysql数据库文件挂载至了本项目的mysql自文件夹**，这种方式也防止因不熟悉docker-compse导致了数据的丢失。

```
sudo rm start-bolo-with-docker-compose -rf
```

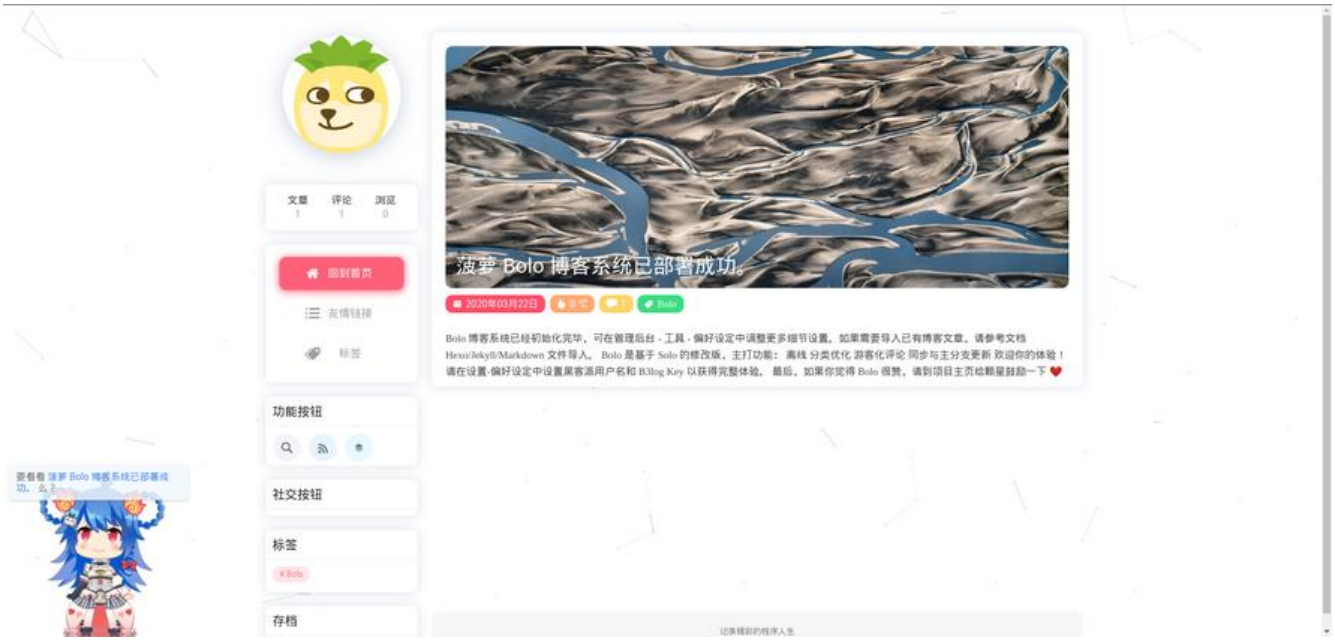
访问测试

再确认已经启动完成之后、使用浏览器访问您设置的对应域名即可完成博客的初始化。

- bolo 初始化界面



- bolo 初始化完成界面



项目介绍

文件结构

```
├── .gitignore
├── LICENSE
├── mysql # mysql 数据库
│   └── data
├── .mysql.env # mysql 容器环境变量配置文件
├── nginx
│   ├── conf.d # nginx 子配置文件目录、可添加自定义配置文件（以.conf结尾）
│   ├── nginx.conf
│   └── ssl
├── README.md
```

原文链接：使用 docker-compose 一键启动 bolo 博客

```
├── .bolo.env # bolo 容器环境变量配置文件
├── theme # 主题文件存放路径、如需挂载自定义主题、请在 docker-compose.yaml 中做好相应
置
│   └── solo-nexmoe
└── web
    └── markdowns # markdown 文件存放路径（使用markdown 文件初始化时bolo使用）详情参
 solo 导入markdown文件
```

docker-compose.yaml

```
version: "3"

services:
  nginx:
    image: nginx:latest
    restart: always
    container_name: "solo-nginx"
    ports:
      - "80:80"
      # - "443:443"
    depends_on:
      - "mysql"
    links:
      - "bolo:bolo"
    volumes:
      - ./nginx/conf.d:/etc/nginx/conf.d:ro
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
    networks:
      - bolo-net

  mysql:
    image: mysql:5.5
    restart: always
    container_name: "solo-mysql"
    expose:
      - "3306"
    volumes:
      - ./mysql/data:/var/lib/mysql
    env_file:
      - ./.mysql.env
    networks:
      - bolo-net
    command: --max_allowed_packet=32505856 --character-set-server=utf8mb4 --collation-se
ver=utf8mb4_general_ci

  bolo:
    image: tangcuyu/bolo-solo:latest
    restart: always
    container_name: "bolo"
    expose:
      - "8080"
    depends_on:
      - "mysql"
    links:
```

```
    - "mysql:mysql"
  # 主题与文章挂载目录
  # volumes:
  #   - ./web/markdowns:/opt/solo/markdowns:rw
  #   - ./theme/solo-nexmoe:/opt/solo/skins/nexmoe
  env_file:
    - ./.bolo.env
  networks:
    - bolo-net
  command: --listen_port=8080 --server_scheme=http --server_host=localhost --server_port


networks:
  bolo-net:
```

mysql 容器环境变量

下面是默认的配置、可根据个人需要进行更该、但需要确保与bolo容器的环境变量值一致、保证数据连接的有效性。

```
# cat .mysql.env

MYSQL_ROOT_PASSWORD=expoli
MYSQL_USER=solo
MYSQL_DATABASE=solo
MYSQL_PASSWORD=solo123456
```

bolo 容器环境变量

下面是默认的配置、可根据个人需要进行更该、但需要确保与bolo容器的环境变量值一致、保证数据连接的有效性。

```
# cat .bolo.env

RUNTIME_DB=MYSQL
JDBC_USERNAME=solo
JDBC_PASSWORD=solo123456
JDBC_DRIVER=com.mysql.cj.jdbc.Driver
JDBC_URL=jdbc:mysql://mysql:3306/solo?useUnicode=yes&characterEncoding=UTF-8&useSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC
```