链滴

个人 solo 博客搭建

作者：NekoChips

原文链接：https://ld246.com/article/1573436158485

来源网站：链滴

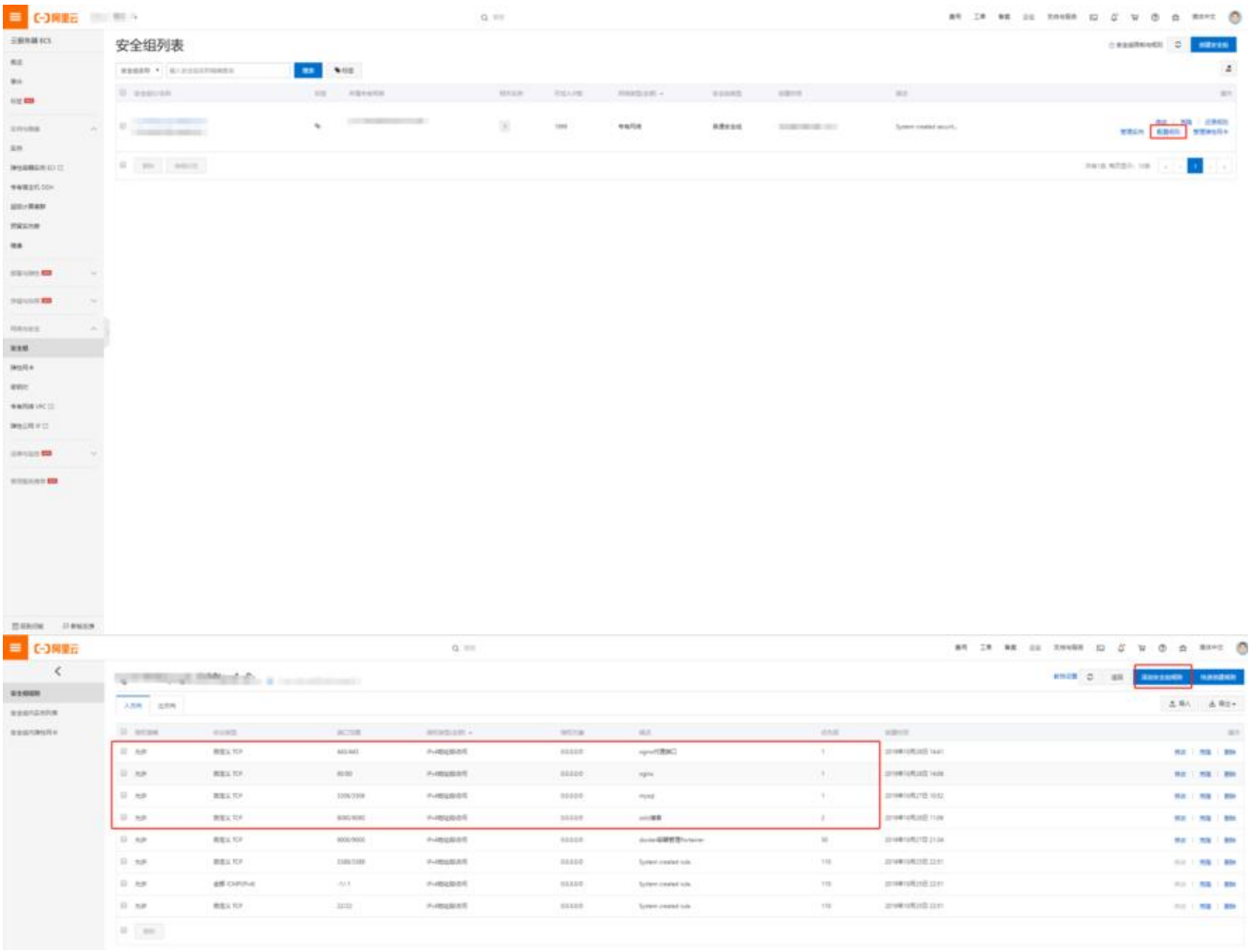
很早之前就开始关注 solo 了，非常喜欢这套精美的个人博客。所以当时将项目 Git 到本地搭建了一本地的环境，出于工作原因也没有太多的时间去整理和记录一些东西到博客中来。这段时间计划把这博客搞起来，以后多记录些用过的技术做回顾参考用。下面就我个人搭建 solo 的过程以及其中遇到问题做个记录。

1.准备工作

当然，如果只是单纯的想了解 solo 博客的话，可以在本地搭建一套环境即可使用。

1.1 服务器

最近阿里云在做活动，优惠力度非常大，可以购买一个自己觉得合适的服务器。配置的话如果只挂 Soo，买最低配 1G1 核 1M 即可。由于搭建 solo 需要用到 docker，建议在选择服务器的系统版本时选 CentOS7.0 以上的版本。购买服务器后，在服务器的安全组中添加配置规则。开放如下端口：80 443、3306、8080。

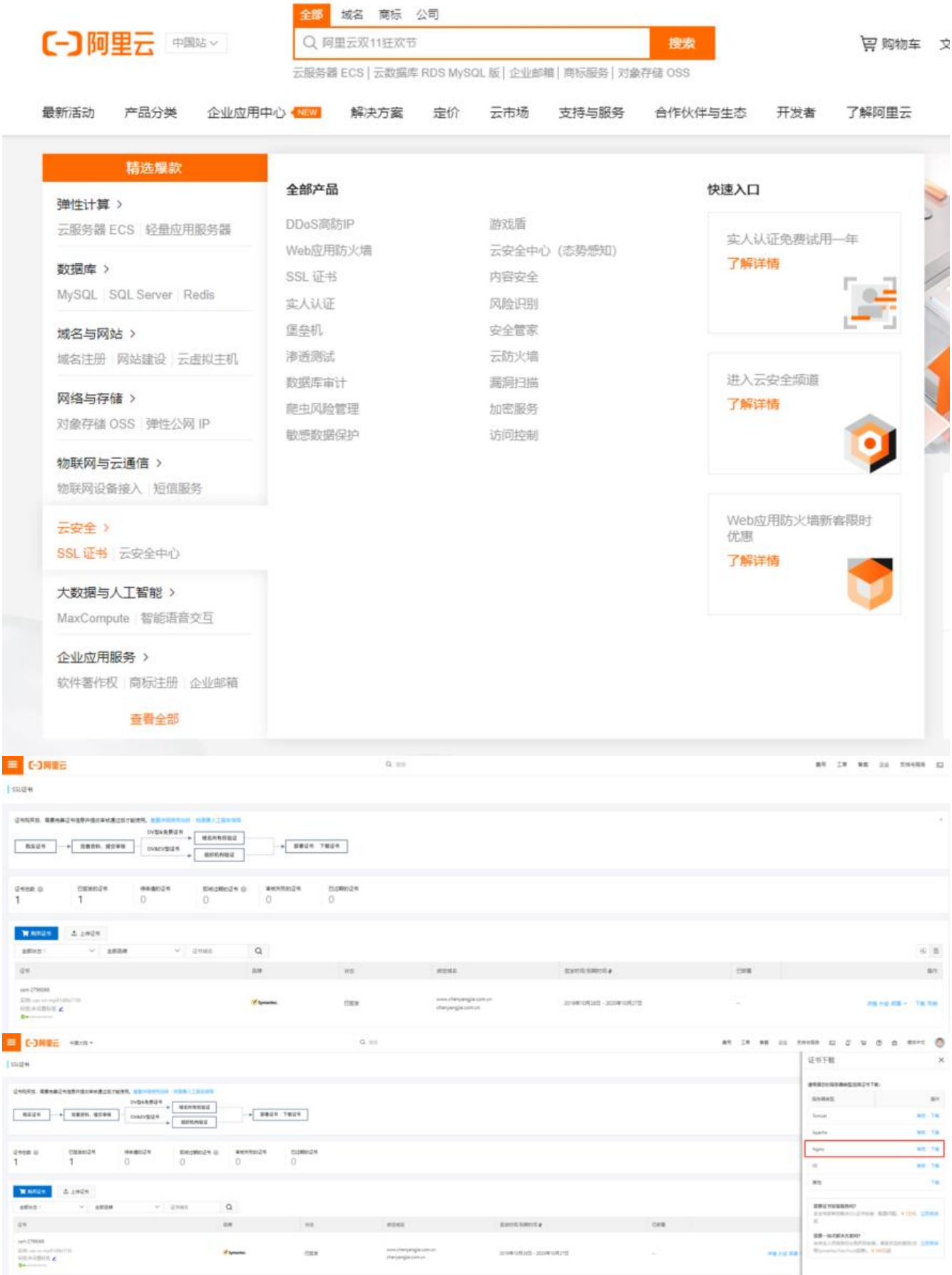


1.2 域名

比起使用 ip 访问来讲，使用域名来访问的会显得比较专业。买域名可选择的服务商有很多，如果服器是购买的阿里云的，这里域名也是首推阿里云，在后面的备案会比较方便。然后和服务器一起便于理。

购买域名时挑选自己喜欢的就行。也是比较幸运，我买的时候发现我的。cn 域名还在。

购买域名后，域名需要进行 实名认证 ---> DNS 解析 ---> 域名备案。全部搞定之后，域名才能正常使用。建议使用 https 安全协议来访问域名，还是那句话“显得更加专业”。这个时候就需要去申请 SL 证书了。申请成功后即可下载对应的 SSL 证书，这里我们选择 Nginx。



PS：DNS 解析需要新增解析记录，一般进入之后会有指引按钮，点击创建即可。这一步没有做的话会导致之后的 nginx 无法解析域名从而启动失败。

2. 安装部署

准备工作都完成了之后，就可以进行 solo 博客的安装部署环节了。

2.1 Docker（已经安装 Docker 的可跳过该环节）

首先安装 Docker，前面要求系统版本一定是要在 CentOS7.0 以上的版本，就是因为 Docker 的环境须使用高于 3.10 的内核版本，否则 Docker 无法正常运行。

- 安装依赖包

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

- 设置阿里云镜像源

```
sudo yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

- 安装 Docker

```
sudo yum install docker-ce
```

- 启动 Docker

```
# 开机自启
sudo systemctl enable docker
# 启动docker服务
sudo systemctl start docker
```

- 验证 Docker 是否安装成功

```
docker version
```

- 设置 Docker 国内镜像源

比较常用的有网易的镜像中心和 daocloud 镜像市场，个人建议使用 daocloud，原因不做阐述。

创建或修改配置文件

```
vi /etc/docker/daemon.json
```

```
{
    "registry-mirrors": ["http://ef017c13.m.daocloud.io"],
    "live-restore": true
}
```

也可以使用命令直接更改（建议使用）

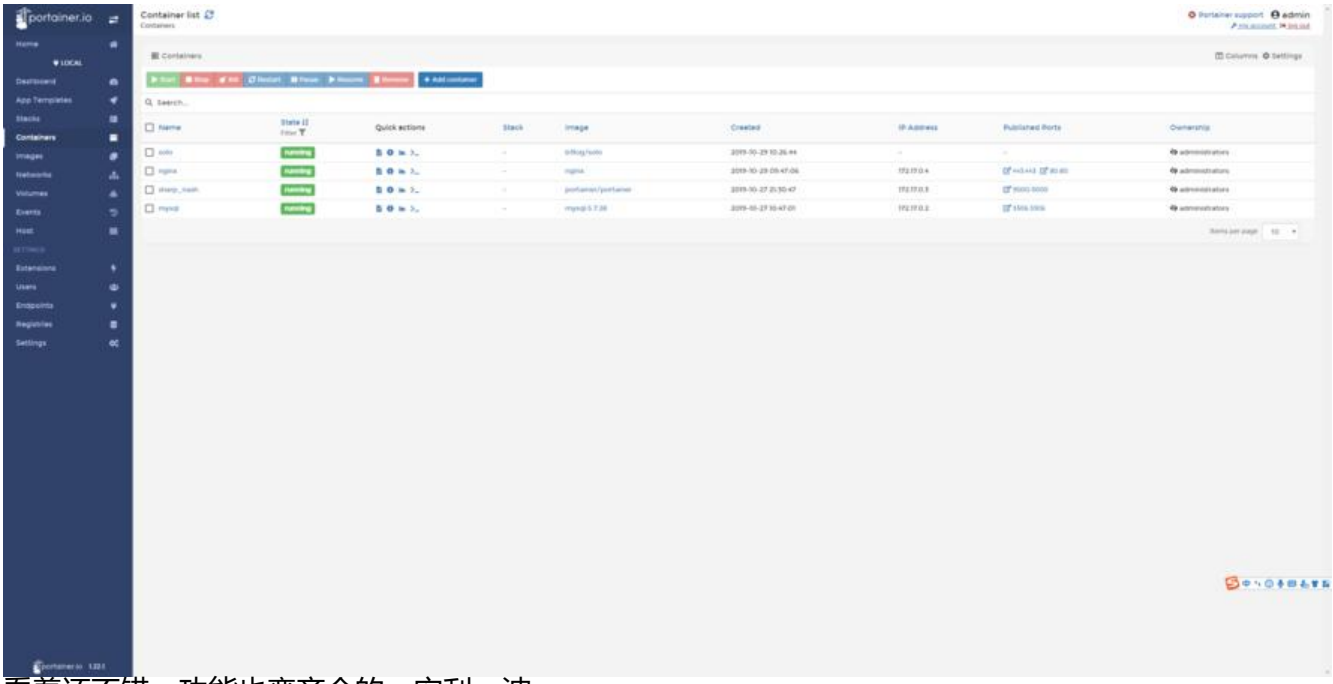
```
curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s http://f1361db2.m.daocloud.io
```

- 重启 Docker

```
sudo systemctl restart docker
```

- 安装 Docker 可视化管理界面 GUI（可选）

```
docker volume create portainer_data
docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer
```



看着还不错，功能也蛮齐全的，安利一波。

至此 Docker 安装配置完成

2.2 安装 MySQL

MySQL 最新版本为 8.X，建议安装较为稳定的 5.6 或 5.7 的版本。这里我选择的是 Mysql5.7.28。

- 拉取镜像

```
docker pull mysql:5.7.28
```

- 启动 MySQL 容器

```
docker run --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=yourpassword -d mysq:5.7.28
```

- 查看容器是否启动成功（可通过可视化界面查看）

```
docker ps -a
```

- 启动成功后进入 MySQL 容器

```
docker exec -it mysql bash
```

- 登录 root 用户并创建 solo 用户和数据库（ 这里不推荐直接使用 root 用户，原因不做阐述。）

```
# 进入数据库 p后面跟你的密码
mysql -uroot -pyourpassword
# 创建数据库(数据库名:solo;字符集utf8mb4;排序规则utf8mb4_general_ci)
create database solo DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
# 创建solo用户
CREATE USER 'solo'@'ip' IDENTIFIED BY 'password';
# 赋予solo用户solo数据库的所有权限
GRANT ALL ON solo.* to 'solo'@'ip' IDENTIFIED BY 'password';
flush privileges;
# 出现Query OK, 1 row affected (0.00 sec)表示成功
#退出数据库
exit
#退出容器
exit
```

至此 MySQL 安装配置完成。

2.3 安装 Solo

终于到主角了。直接运行以下命令会自动 pull solo 的镜像并启动。

```
docker run --detach --name solo --network=host \
--env RUNTIME_DB="MYSQL" \
--env JDBC_USERNAME="****" \
--env JDBC_PASSWORD="******" \
--env JDBC_DRIVER="com.mysql.cj.jdbc.Driver" \
--env JDBC_URL="jdbc:mysql://127.0.0.1:3306/solo?useUnicode=yes&characterEncoding=UT
-8&useSSL=false&serverTimezone=UTC" \
b3log/solo --listen_port=8080 --server_scheme=http --server_host=www.chenyangjie.com.cn
```

参数说明：

- --env RUNTIME_DB="MYSQL" 运行时数据库类型
- --env JDBC_USERNAME="solo" 数据库用户名称 solo
- --env JDBC_PASSWORD="password" 数据库用户 solo 的密码
- --env JDBC_DRIVER="com.mysql.cj.jdbc.Driver" \ 数据库驱动
- --listen_port=8080 监听端口为 8080
- --server_scheme=http 访问协议为 http
- --server_host=www.chenyangjie.com.cn 访问地址，有域名的写域名，没域名的写服务器外网 ip 地址

启动成功后即可通过域名或 ip:8080 访问 solo 博客了。首次访问 solo 博客会进行初始化。初始化完后即可使用，使用方法参照 Solo 用户指南

2.4 安装 Nginx

Nginx 这块我不太熟，这里参照下大佬的安装过程。建议查看从零开始安装 solo 博客

```
# 切换到服务器根目录
cd /
# 创建主目录
mkdir dockerData
# 创建文件
mkdir dockerData/nginx dockerData/nginx/conf dockerData/nginx/logs dockerData/nginx/w
w dockerData/nginx/ssl
```

- dockerData/nginx/conf 存放 nginx 相关配置文件
- dockerData/nginx/logs 存放 nginx 日志文件
- dockerData/nginx/www 存放 nginx 访问的资源文件
- dockerData/nginx/ssl 存放 ssl 证书

启动 Nginx

```
docker run --name nginx -p 80:80 -d --rm nginx
```

Nginx 启动成功并能正常访问后导出配置文件

- docker cp nginx:/etc/nginx/nginx.conf /dockerData/nginx/conf/nginx.conf 导出配置文件 ngi
x.conf
- docker cp nginx:/etc/nginx/conf.d /dockerData/nginx/conf/conf.d 导出配置为你 nginx.conf

停止 Nginx 服务

```
docker stop nginx
# 会自动删除现在的 nginx 容器，然后执行如下命令重新启动一个 nginx 容器
```

将之前下载的 Nginx 的 SSL证书 中的 pem 文件和 key 文件放至 dockerData/nginx/ssl 目录下。

然后配置 Niginx 文件

```
cd /dockerData/nginx/conf/conf.d
vim default.conf
```

```
server {
    listen       443 ssl;
    server_name  localhost;
    # ssl on;

    #charset koi8-r;
    #access_log  /var/log/nginx/host.access.log  main;

    ssl_certificate /ssl/1_******************.pem;  # ssl 证书目录
    ssl_certificate_key /ssl/1_******************.key;
    ssl_session_timeout 5m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE;
    ssl_prefer_server_ciphers on;
```

```
    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }
    # ......
}
server{
  listen 80;
  server_name localhost;
  rewrite ^(.*) https://$host$1 permanent;
}
```

启动新的 Nginx，并使用 https 协议访问

```
docker stop nginx  # 停止容器
docker rm nginx # 删除容器
# 启动新的
docker run -d -p 80:80 -p 443:443 --name nginx \
-v /dockerData/nginx/conf/nginx.conf:/etc/nginx/nginx.conf \
-v /dockerData/nginx/conf/conf.d:/etc/nginx/conf.d \
-v /dockerData/nginx/ssl:/ssl/ \
-v /dockerData/nginx/www:/usr/share/nginx/html \
-v /dockerData/nginx/logs:/var/log/nginx nginx
```

- -p 443:443 监听 443 端口，没进行备案的域名该端口可能访问不到
- -v /dockerData/nginx/ssl:/ssl/ 挂载 ssl 证书目录

2.5 使用 Nginx 反向代理实现对 solo 的访问

停止 solo，并移除 solo 容器（同样可以通过 Portainer 进行操作）

```
docker stop solo
docker rm solo
```

配置 Nginx 配置文件实现 Nginx 反向代理

```
cd /dockerData/nginx/conf/conf.d
vim default.conf

upstream backend {
    server localhost:8080; # Solo 监听端口
}
location / {
        proxy_pass http://www.chenyangjie.com.cn:8080;
    proxy_set_header  Host $http_host;        // proxy_set_header一定要进行配置
        proxy_set_header  X-Real-IP  $remote_addr;    // 不然会造成静态资源访问错误以及可
                        // 能造成“Latke 配置错误”
    }
# 替换上面部分即可
# 按esc，然后输入:wq保持退出
```

重启 solo 容器

```
docker run --detach --name solo --network=host \
--env RUNTIME_DB="MYSQL" \
--env JDBC_USERNAME="****" \
--env JDBC_PASSWORD="******" \
--env JDBC_DRIVER="com.mysql.cj.jdbc.Driver" \
--env JDBC_URL="jdbc:mysql://127.0.0.1:3306/solo?useUnicode=yes&characterEncoding=UT
-8&useSSL=false&serverTimezone=UTC" \
b3log/solo --listen_port=8080 --server_scheme=https --server_host=www.chenyangjie.com.c
 --server_port=
```

- --server_scheme=http 换成 --server_scheme=https 即可
- --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

重启 Nginx

```
docker restart nginx
```

至此个人的 solo 博客就搭建完成了。

3. 其他组件

3.1 挂载第三方皮肤

创建皮肤文件目录，如：/dockerData/solo/skins。将皮肤放到该目录下。

** tips ** Solo 默认使用的皮肤是 “Pinghsu”，务必将该皮肤加入文件目录中。

添加 solo 启动项配置，将皮肤文件目录挂载至 solo

```
# **tips** 这行配置一定要卸载“listen_port=”之前，否则由于无法监听到皮肤导致solo启动失败
--volume /dockerData/solo/skins/:/opt/solo/skins/ \
```

3.2 使用 Lute

Lute 使用指南传送门：Lute 使用指南

Lute 安装指南传送门：安装 Lute-http

添加 solo 启动配置项

```
# **tips** 先启动 lute 再添加下面启动配置启动 solo
--lute_http=http://127.0.0.1:8249 --lute_http=http://localhost:8249 --lute_http=
```

查看 solo 启动日志，启动参数 luteAvailable=true 即为 Lute 插件安装成功。

3.3 Solo 自动更新

Solo 自动更新传送门：Solo 自动更新

4. 后记

该文章仅供他人参考以及自身回顾作用，如有问题请指出，即刻进行修改。

技术参考：

Docker（一）Linux 开启你的 Docker 之旅

linux 安装 docker

从零开始安装 solo 博客