链滴

Solo 搭建个人博客心得（最简版）

作者：StephenZhang

原文链接：https://ld246.com/article/1598764139243

来源网站：链滴

Preface

好容易熬过了域名ICP备案，可以正式开始搭建博客了……

虽然在社区已经有了很多类似的，使用Solo搭建博客的心得，但是我可以说，我这篇心得应该是最清明了适合运维小白的，而且也适于运营维护。为什么呢？因为我也是运维小白，很多时候可以理解看大神们的术语时的痛苦……

0. 前期准备

在正式搭建之前，你要做好如下准备：

- 一台拥有公网IP的Linux服务器（RH/Debian或其衍生发行版均可）
- 一个经过ICP备案的域名（可以解析到你的IP上）
- 在域名提供商那里申请的SSL证书文件，注意选择Nginx版本的下载到本地备用

注：个人不喜欢在写文章时过多的插入图片，因此希望读者能够拥有一些Linux操作的基本能力

1. 方案选择

根据官方指南，最好是使用Docker对Solo进行部署，同时为了方便运营维护，选用MySQL作为数据为好；同时，为了使服务器作用最大化，而不是由Solo独占，可以使用Nginx反向代理以处理对Solo访问请求。

这样的话，显然，使用Docker部署Solo是我们搭建工作的重头戏，Nginx则以“工具人”的身份出现……实际上，Solo框架也可以看做是一个工具人，对我们后续的运维来说，MySQL中的数据才是最重的。

所以在参考了众多文章之后，我最后给出的方案是，Solo和Nginx使用Docker部署，而MySQL则部在本地，这样方便对MySQL的数据库进行备份，也方便Solo和Nginx的更新。

2. 准备一下

首先，使用SSH登录云服务器，更新系统，并安装Docker和MySQL（以Ubuntu Server 20.04 LTS例）：

```
sudo apt update && sudo apt upgrade -y       # 更新系统
sudo apt install docker mysql-server         # 安装docker和MySQL，如果已经安装过可以忽略
```

然后，使用docker启动Nginx，确定80端口是否开放：

```
sudo docker run nginx:latest -d --rm --name=nginx --network=host
```

上面这条命令比较长，基本含义是使用运行最新的nginx镜像。-d是指使用分离模式，即运行在后台；-rm是为了方便待会停止它的时候可以自动删除该实例；--name指定了实例的名称；--network指定网络模式，配置为host的意思是不使用内建的docker网桥，而直接使用主机网络进行通信。

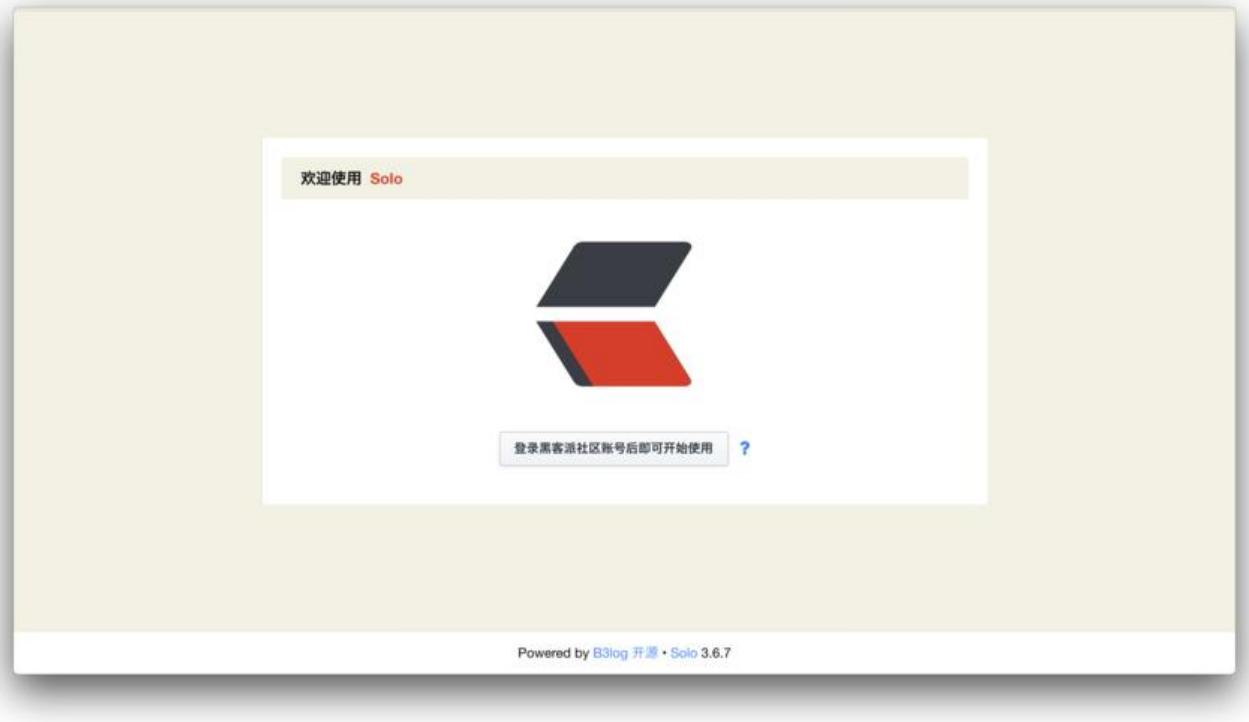
现在可以在浏览器地址栏中输入你的IP地址，如果可以访问到nginx欢迎页面，则证明80端口已经打；否则就要按照你的云服务商的教程打开端口。在我们的配置过程中，需要打开的端口有80（http访）、443（https访问）、3306（数据库访问）以及8080（Solo监听端口，测试用）。你可以一次性它们都打开。

nginx访问成功后，使用sudo docker stop nginx停止并删除该实例。

现在准备启动Solo，这一步可以参考官方教程：启动容器一节。这一步的启动命令较为冗长，要有耐。个人建议命令可以稍微修改一下，加上--rm参数，方便测试完之后删除：

```
sudo docker run --rm --detach --name solo --network=host \
    --env RUNTIME_DB="MYSQL" \
    --env JDBC_USERNAME="root" \
    --env JDBC_PASSWORD="123456" \
    --env JDBC_DRIVER="com.mysql.cj.jdbc.Driver" \
    --env JDBC_URL="jdbc:mysql://127.0.0.1:3306/solo?useUnicode=yes&characterEncoding=
TF-8&useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true" \
    b3log/solo --listen_port=8080 --server_scheme=http --server_host=localhost --server_por
=
```

当Solo的实例启动后，可以通过your_ip_addr:8080查看是否启动成功，如果看到类似下图的开始使界面，则证明成功：



现在删除Solo的实例，在云服务器设置中关闭8080端口，现在已经不再需要它了。

3. 正式搭建

上一步中的所有内容均可以看作测试，现在我们要正式开始搭建博客了。

在你的家目录~下新建一个文件夹，文件夹的结构如下：

```
Blogs
├── backups
│   └── conf
│       └── solo.conf
├── nginx
```

```
│   ├── conf.d
│   │   └── solo.conf
│   └── ssl
│       ├── stephen_zhang.key
│       └── stephen_zhang.pem
```

其中，backups文件夹下是对配置文件的备份，后续还可以存放数据库的备份文件；nginx文件夹下在两个子文件夹，分别是conf.d，对应/etc/nginx/conf.d，而ssl/存放的是我们最开始下载好的SSL书文件。nginx文件夹将会被挂载到docker中的nginx实例中去。

BTW，证书一般分为两个文件，其中.pem文件和.crt文件等价。

现在来看solo.conf文件：

```
upstream backend {
    server localhost:8080; # Solo 监听端口（不要修改）
}

server {
    listen 443 ssl;
    server_name localhost;
    access_log off;

    ssl_certificate      /ssl/stephen_zhang.pem;  # 这里修改为你的证书文件名，路径不要修改
    ssl_certificate_key  /ssl/stephen_zhang.key;  # 同上
    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 / {
        proxy_pass http://backend$request_uri;
        proxy_set_header  Host $http_host;
        proxy_set_header  X-Real-IP $remote_addr;
        client_max_body_size  10m;
    }
}

server {
    listen 80 default; # default是必须的，否则通过IP访问时只会看到nginx欢迎页面
    server_name  stephen-zhang.cn; # 修改为你的域名
    rewrite ^(.*) https://$server_name$1 permanent;
}
```

按照以上示例，把修改后的solo.conf保存好，备用。

为了使Solo可以渲染更多的Markdown语法，例如LaTeX数学公式，脚注等等，我们需要启用lute-http：

```
sudo docker run --detach --name lute \
        --network=host \
        --rm b3log/lute-http:latest
```

然后把上一步中Solo的启动命令加以修改，使之可以访问lute（默认端口为8249，无需修改，也不必放此端口）即可：

```
sudo docker run --rm --detach --name solo --network=host \
    --env RUNTIME_DB="MYSQL" \
    --env JDBC_USERNAME="root" \
    --env JDBC_PASSWORD="solo19981120" \
    --env JDBC_DRIVER="com.mysql.cj.jdbc.Driver" \
    --env JDBC_URL="jdbc:mysql://127.0.0.1:3306/solo?useUnicode=yes&characterEncoding=
TF-8&useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true" \
    b3log/solo:latest --listen_port=8080 \
    --server_scheme=https --server_host=localhost --server_port= \
    --lute_http=http://localhost:8249
```

当然了，我们要为自己的博客配置HTTPS，那么上述命令中的--server_scheme的值必须是https，则通过HTTPS访问时会产生异常。

最后，启动nginx反代即可：

```
sudo docker run --detach --rm --name nginx \
        --volume ~/Blogs/nginx/ssl/:/ssl/ \
        --volume ~/Blogs/nginx/conf.d/solo.conf:/etc/nginx/conf.d/solo.conf \
        --network=host nginx:latest
```

这条命令使用--volume参数将我们自己的ssl/挂载到了docker中的/ssl/下，所以solo.conf中证书的径只能是/ssl/....；同时它也用该参数将solo.conf挂载到了docker中的/etc/nginx/conf.d/solo.conf，即nginx实例使用该配置文件进行反向代理。

现在，可以通过如下的方式访问你的博客了：

- 域名： http://your_domain_name或者https://your_domain_name均可
- IP地址： http://your_ip_addr

注：证书是和域名绑定的，因此不要用https://your_ip_addr的方式去访问，否则浏览器会报证书错误异常。

到此，博客的搭建就结束了，你可以把我们上面用到的命令保存为bash脚本，方便重启博客时使用。

4. 优化

优化主要针对两种情况：一种是云服务器带宽太小，另一种是云服务器内存太小。

当带宽小时，我们可以使用CDN加速器，对博客的一些公有资源进行加速，例如内置的皮肤资源（对三方皮肤无效）。这时，只需要在启动Solo实例的命令后添加如下选项并重启即可：

```
--static_server_scheme=https
--static_server_host=cdn.jsdelivr.net
--static_server_port=
--static_path=/gh/88250/solo/src/main/resources
```

对于服务器内存太小，无法支撑MySQL的消耗的情况，可以设法对MySQL进行配置，或者选用H2 Database作为默认的数据库。