Centos7&&NetCore

参考：<https://www.cnblogs.com/A00000/p/10601196.html>

## [CentOS7安装.Net Core2.2](https://www.cnblogs.com/A00000/p/10601196.html)

1、需装Dotnet Core Runtime，执行下列命令，每台电脑一次；

sudo rpm -Uvh <https://packages.microsoft.com/config/rhel/7/packages-microsoft-prod.rpm>

Step 2：安装Dotnet Core Runtime，执行下列命令

sudo yum update

sudo yum install aspnetcore-runtime-2.2

3、可选：如果想要在Linux做 .NET Core的开发和编译工作，那么需要安装 Dotnet Core SDK。Dotnet Core SDK中包括了Dotnet Core Runtime；

sudo yum update

sudo yum install dotnet-sdk-2.2

4、查看版本

Dotnet --version

5、选做：使NetCore的端口在Centos7中暴露外环境，通过IP可以访问；

5.1代码写端口

public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>

WebHost.CreateDefaultBuilder(args)

.UseStartup<Startup>()

.UseUrls("http://\*:5000");

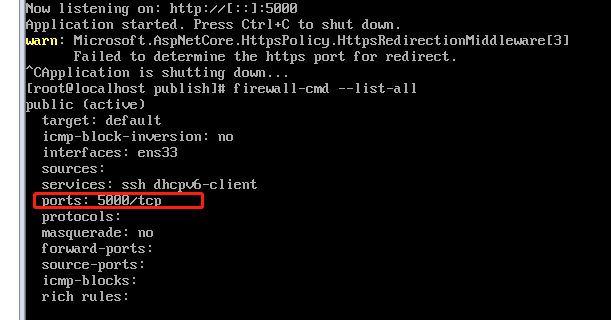
5.2 Centos7开发端口：

firewall-cmd --state ##查看防火墙状态，是否是running

firewall-cmd --add-port=5000/tcp –-permanent ### >>>开启端口

firewall-cmd –-reload ### >>>重启防火墙

firewall-cmd --list-all ###查看列表



## 5.3 作为服务运行（参考）

<https://www.cnblogs.com/imstrive/p/9674576.html>

6、使用Nginx反向代理（不使用开放防火墙的方法）

因为Kestrel是一个很基本的web服务器，所以需要配置性能更好的IIS、Nginx或者Apache。在Kestrel前置Apache服务器，Apache可以作为Kestrel的反向代理。这样可以通过反向代理将部分链接指向ASP.NET Core，而部分代码仍旧使用php或者java，为程序的迁移提供了便利。

Centos7&&Nginx

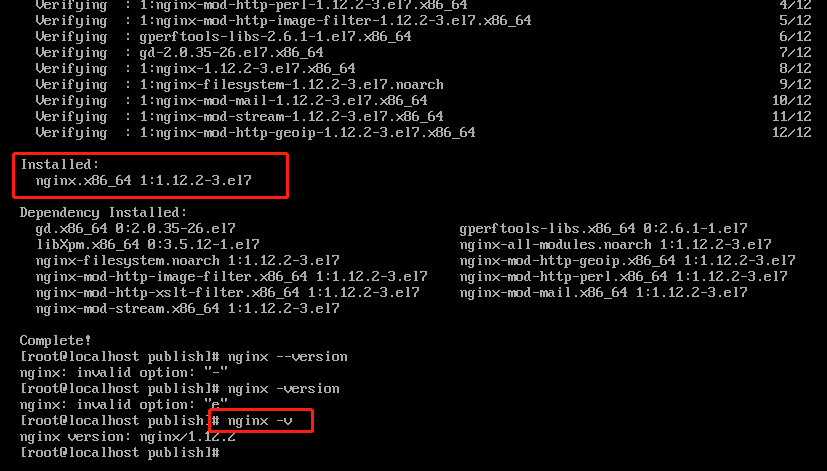
参考<https://blog.csdn.net/u013264030/article/details/79853614>

# 图解在CentOS7配置nginx服务(支持多域名，禁IP直访，防恶意解析)

1. 安装Nginx

Yum search nginx ###查看Yum 源中包含Nginx

Sudo yum install –y nginx ###安装Nginx



1. nginx -t 可用来检查一遍是否配置脚本运行正常



3、启动Nginx并设置开机自动运行

sudo systemctl start nginx.service

sudo systemctl enable nginx.servic

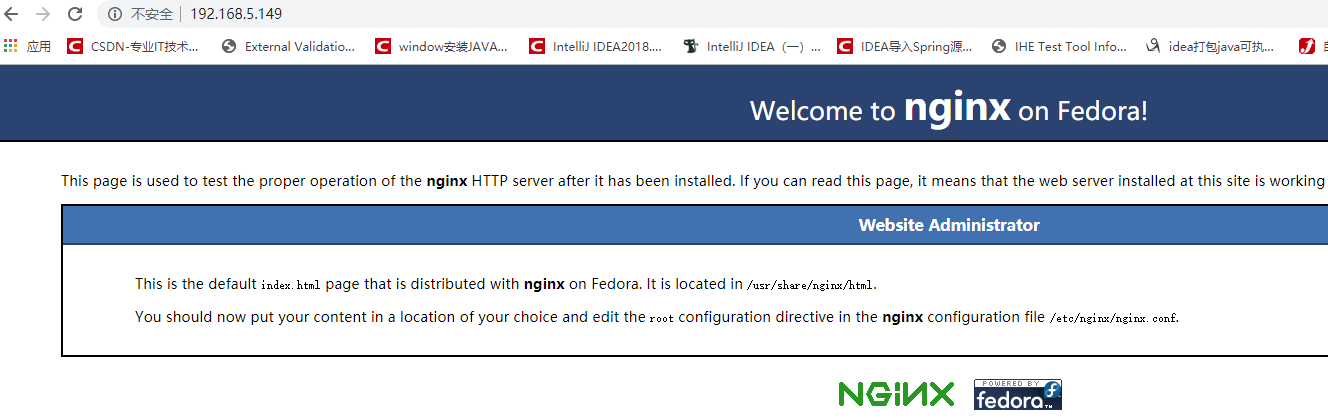
4、Centos7开放端口：

firewall-cmd --state ##查看防火墙状态，是否是running

firewall-cmd --add-port=80/tcp –-permanent ### >>>开启端口

firewall-cmd –-reload ### >>>重启防火墙

5、浏览<http://xx.x.x.x:80>打开Nginx网站



6）关闭SELinux ,使用getenforce 查询

查看当前selinux的状态。

  /usr/sbin/sestatus

修改配置文件需要重启机器：

　　　　　　修改/etc/selinux/config 文件

　　　　　　将SELINUX=enforcing改为SELINUX=disabled

　　　　　　重启机器即可

6）将nginx.config的user改为和启动用户一致，

命令：vi conf/nginx.conf



6) 修改Nginx 配制，保存

              命令: vim /etc/nginx/nginx.conf

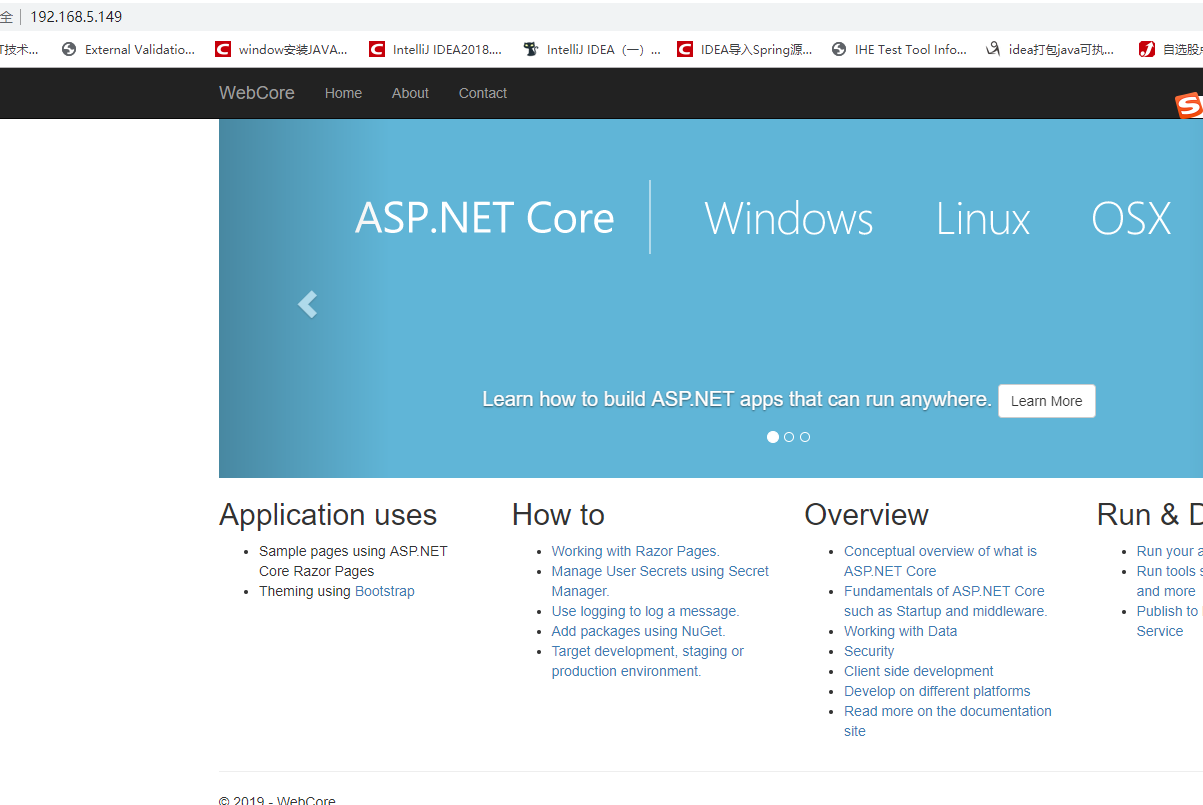
server {  
  listen 80;  
  location / {  
    proxy\_pass http://localhost:5000;  
  }  
}

7) 重新加载Nginx配制文件

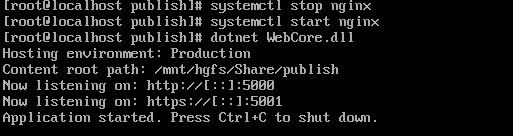
             命令：nginx -t 测试配制文件是否正确

             命令：nginx -s reload 重新加载nginx配制文件，不用重启nginx

8）浏览<http://xx.x.x.x:80>，就会打开重新定向的网站



9）Web网站要在Centos7上使用HTTPS ，必须要安装SDK，还要系统注册证书：dotnet dev-certs https *--trust*



题外话：

<https://www.cnblogs.com/A00000/p/10598844.html>

## [Docker学习系列（一）-CentOS7下安装Docker](https://www.cnblogs.com/A00000/p/10598844.html)

NetCore&&Docker

[https://hub.docker.com](https://hub.docker.com/) Docker仓库

1. 拉取Microsoft/dotnet镜像

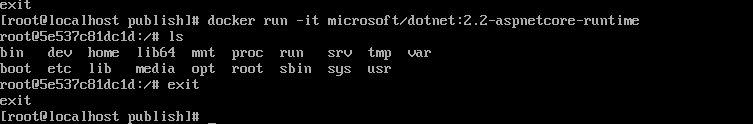
Docker pull Microsoft/dotnet:2.2-aspnetcore-runtime (只有运行时)

Docker pull Microsoft/dotnet (SDK)

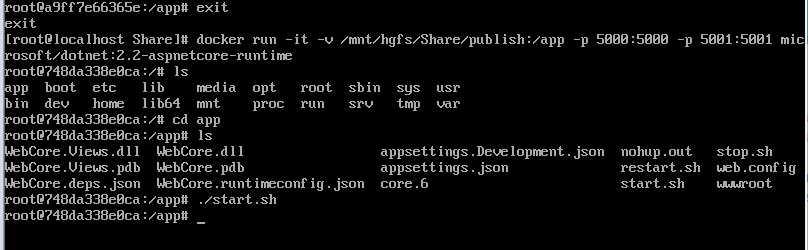
2、进入镜像内，查看内容

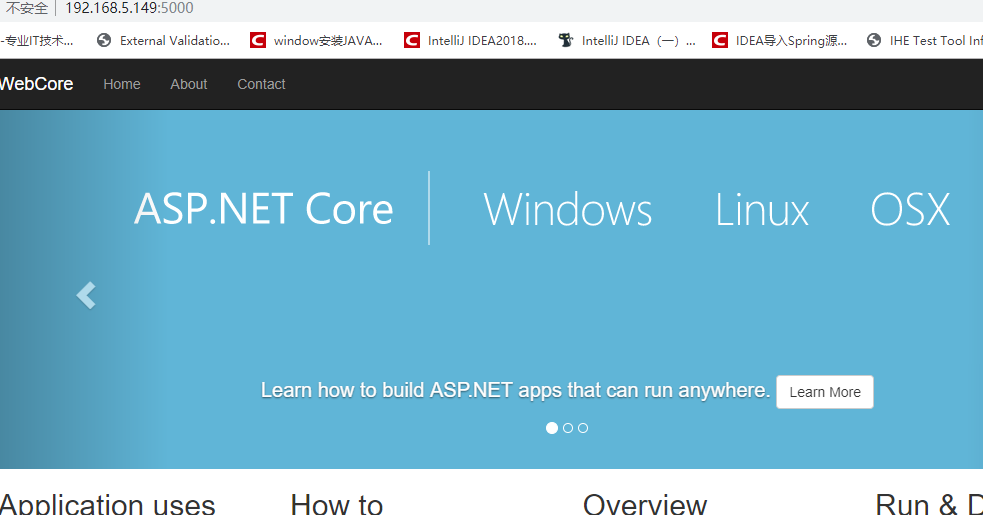
docker run –it Microsoft/dotnet:2.2-aspnetcore-runtime

-it 交互式进入Docker



3、（可选）挂载宿主机的项目目录到容器内

Docker run –it –v /mnt/hgfs/Share/publish:/app –p 5000:5000 -p 5001:5001 microsoft/dotnet:2.2-aspnetcore-runtime 



1. 使用Dockerfile(不再使用步骤3)

Docker build –t coreweb:v1.0 .

Docker run –d –p 5000:5000 -v /var/log/webcore/t1:/app/logfile coreweb:v1.0

-d:容器在后台运行

-p:将容器内部的接口映射到我们使用的主机上

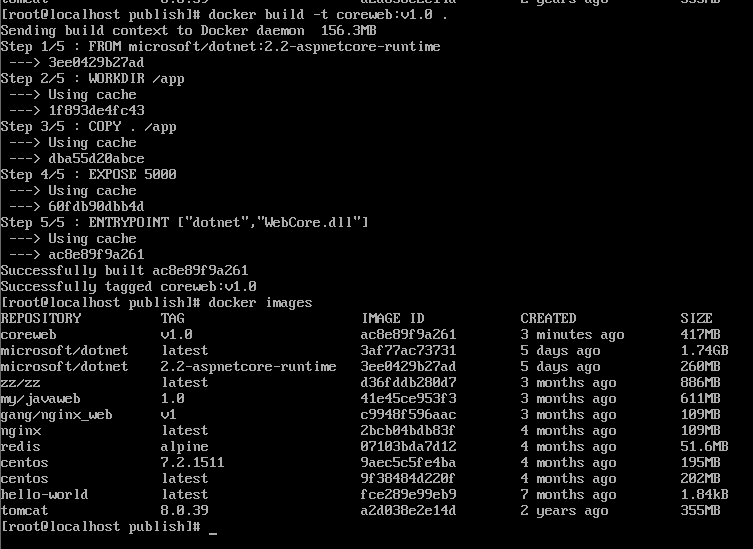
--name:给容器指定名字（可无）

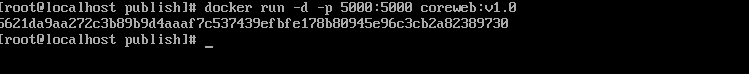
**查看端口；**(使用容器ID 或者 容器名称)

docker port ID/NAMES

**查看容器运行日志**；检查WEB应用程序

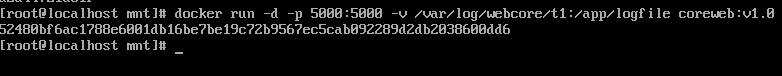
docker logs -f ID

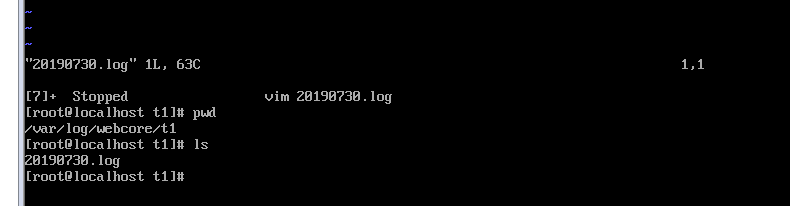


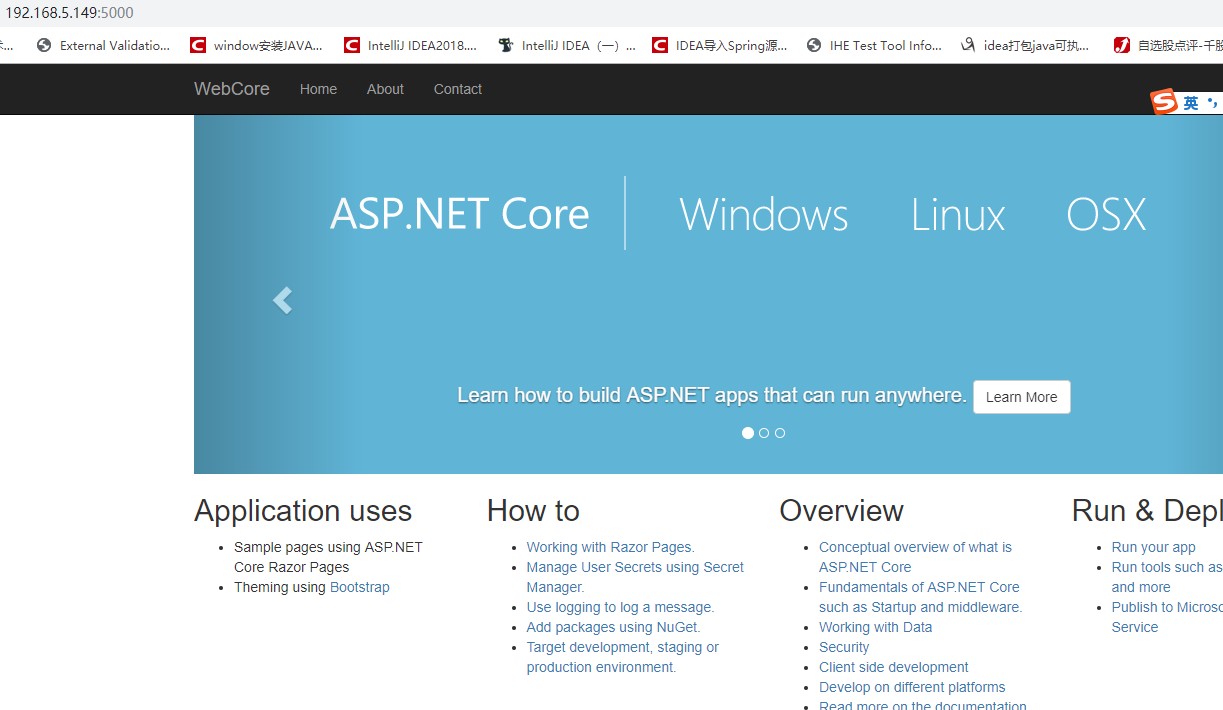


带有Log的卷指向：概念参考：<https://www.cnblogs.com/reachos/p/8621748.html>

Docker run –d -p 5000:5000 –v /var/log/webcore/t1:/app/logfile coreweb:v1.0







1. 删除容器



进阶学习：

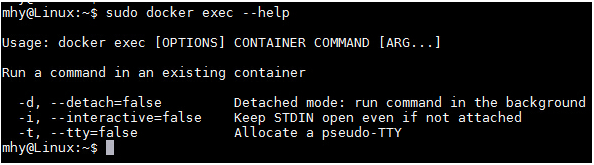
docker inspect test1/Id | more

查看容器详情；

**（**额外）使用docker exec进入Docker容器

　　除了上面几种做法之外，docker在1.3.X版本之后还提供了一个新的命令exec用于进入容器，这种方式相对更简单一些，下面我们来看一下该命令的使用：

1. $ sudo docker exec --help



接下来我们使用该命令进入一个已经在运行的容器

1. $ sudo docker ps
2. $ sudo docker exec -it 775c7c9ee1e1 /bin/bash

Docker&&Nginx

1. 拉取Nginx(laster)

Docker pull nginx

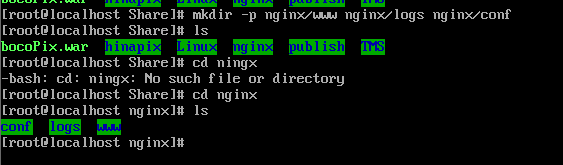
1. 新建目录nginx,用于存放Nginx的持久化内容；参考代码工程的Doc/nginx/

Mkdir –p nginx/www nginx/logs nginx/conf

-p 递归新建，哪一级目录没有，就新建  
www:目录将映射为nginx容器配置的虚拟目录

logs：目录将映射为nginx容器的日志目录

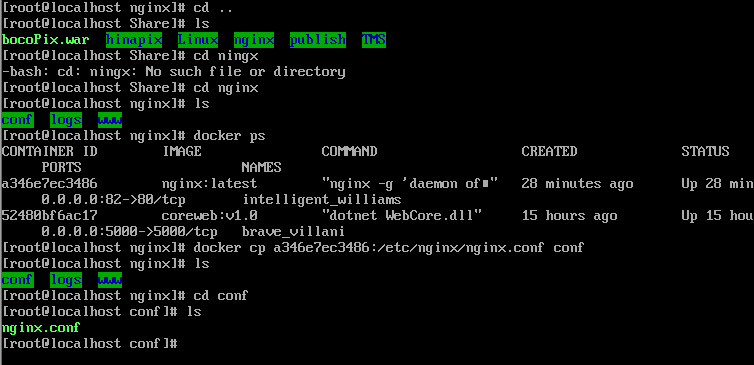
conf: 目录的配置文件将映射为Nginx容器的配置文件



1. 将容器内的Nginx的配置文件拷贝到目录Conf

Docker ps –a

Docker cp id:/etc/nginx/nginx.conf conf

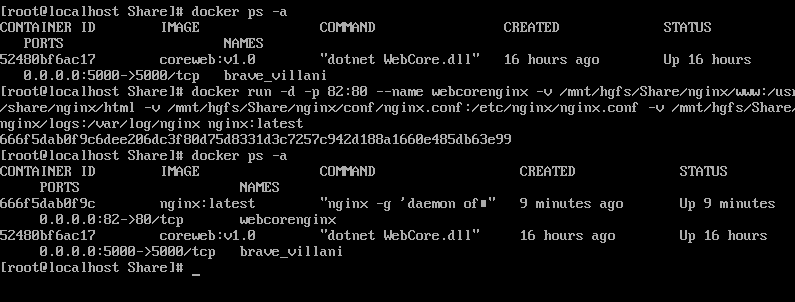


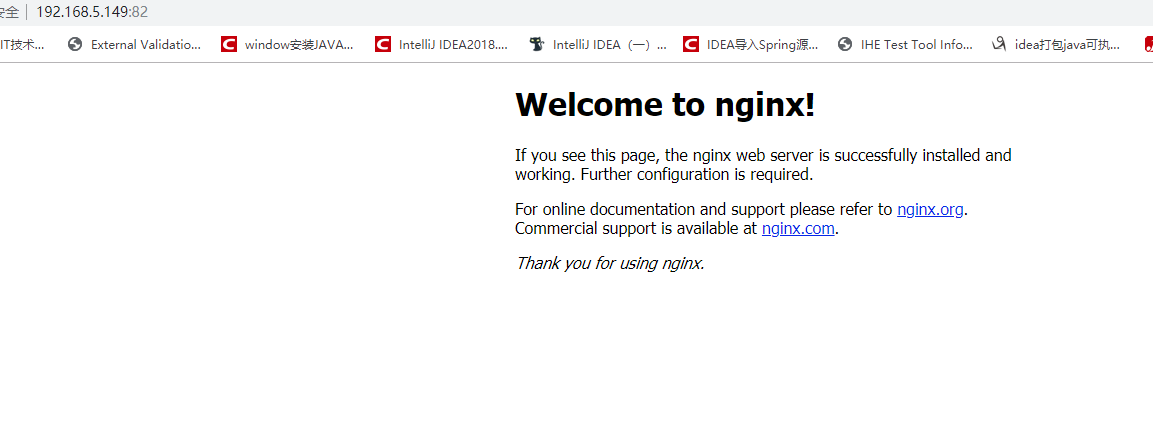
1. 部署

Docer run –d -p 82:80 --name webcorenginx -v /mnt/hgfs/Share/nginx/www:/usr/share/nginx/html -v /mnt/hgfs/Share/nginx/conf/nginx.conf:/etc/nginx/nginx.conf -v /mnt/hgfs/Share/nginx/log:/var/log/nginx nginx:latest

注意外层映射的目录将完全替代容器中的目录，容器目录中有的文件而外层目录没有，映射后，则容器得不到原来容器的目录文件；

外层映射的文件也将完全代替容器中的文件：





### 相关命令

如果要重新载入 NGINX 可以使用以下命令发送 HUP 信号到容器：

$ docker kill -s HUP container-name

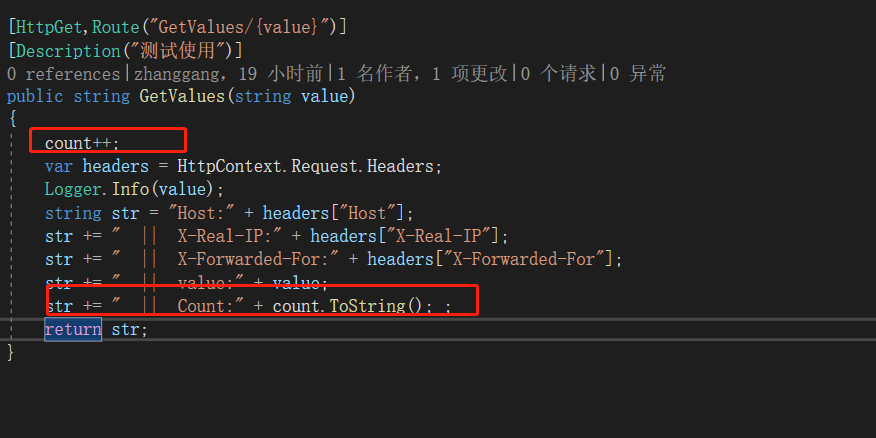
重启 NGINX 容器命令：

$ docker restart container-name

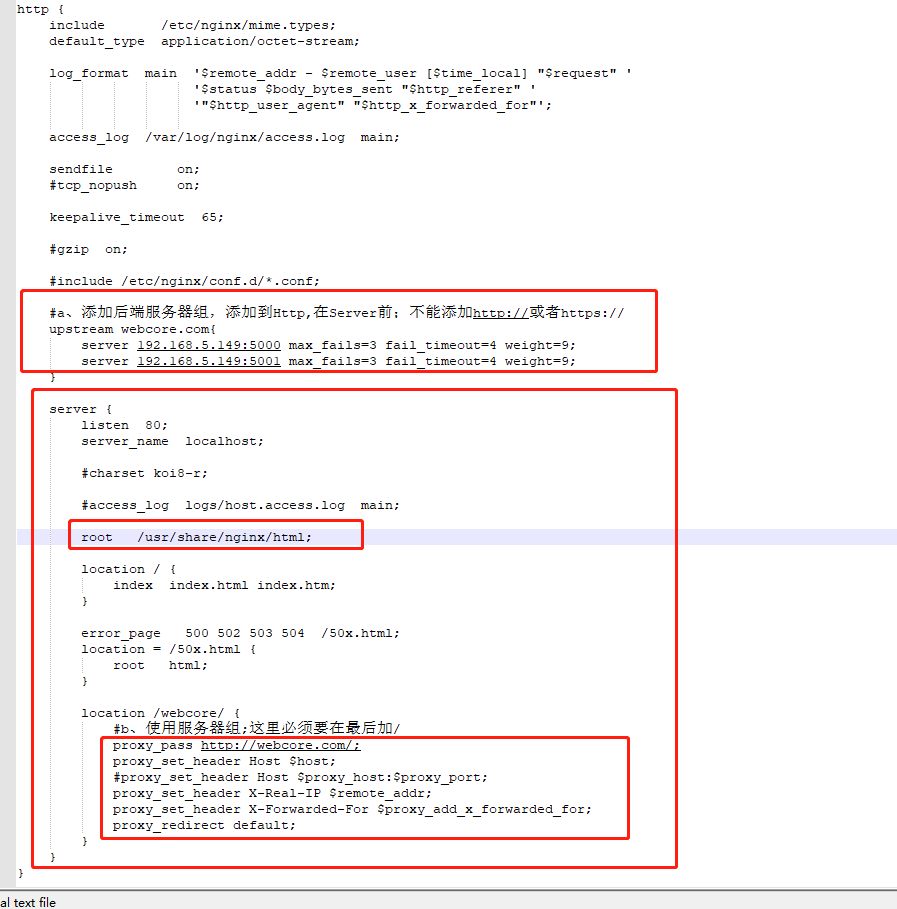
多WebApi Docker（同镜像）部署，使用Nginx负载：

<http://192.168.5.149:82/webcore/api/values/GetValues/ccccFT>；

* 1. Webapi 代码改造，方便知道是否负载均衡



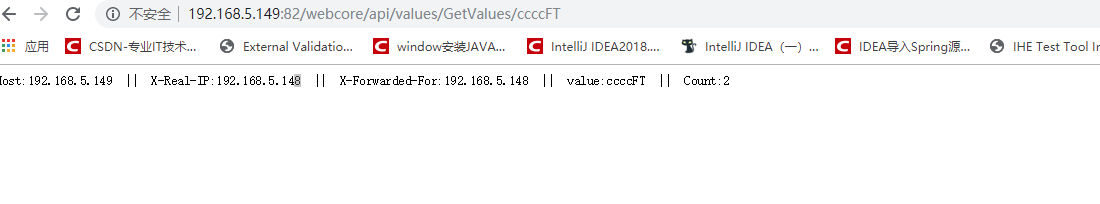
* 1. ngxin.conf 改变，此处采时间片轮询机制



* 1. 启动两个WebApi 容器：
  2. 启动一个Nginx（反向代理）容器



* 1. 在客户端机器上访问：<http://192.168.5.149:82/webcore/api/values/GetValues/ccccFT>



* 1. 完成

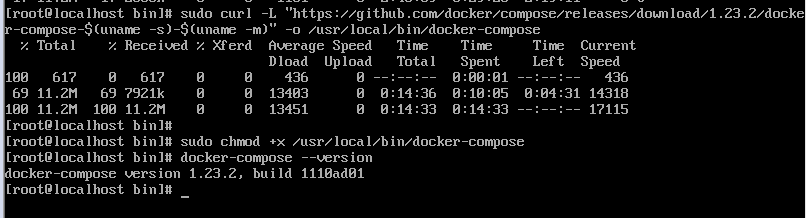
Docker && docker-compose

1. 安装docker-compose

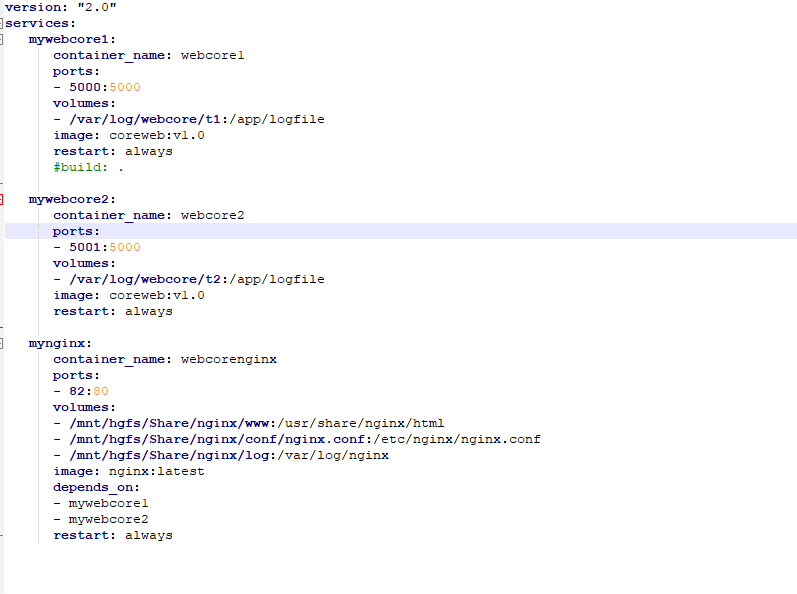
sudo curl -L "https://[github](https://www.centos.bz/tag/github/).com/docker/compose/releases/download/1.23.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version

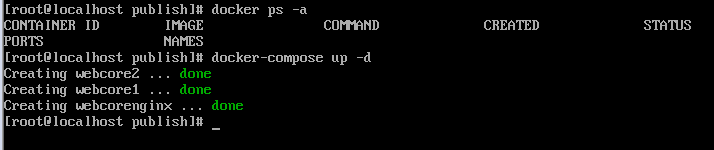


1. 进行编排

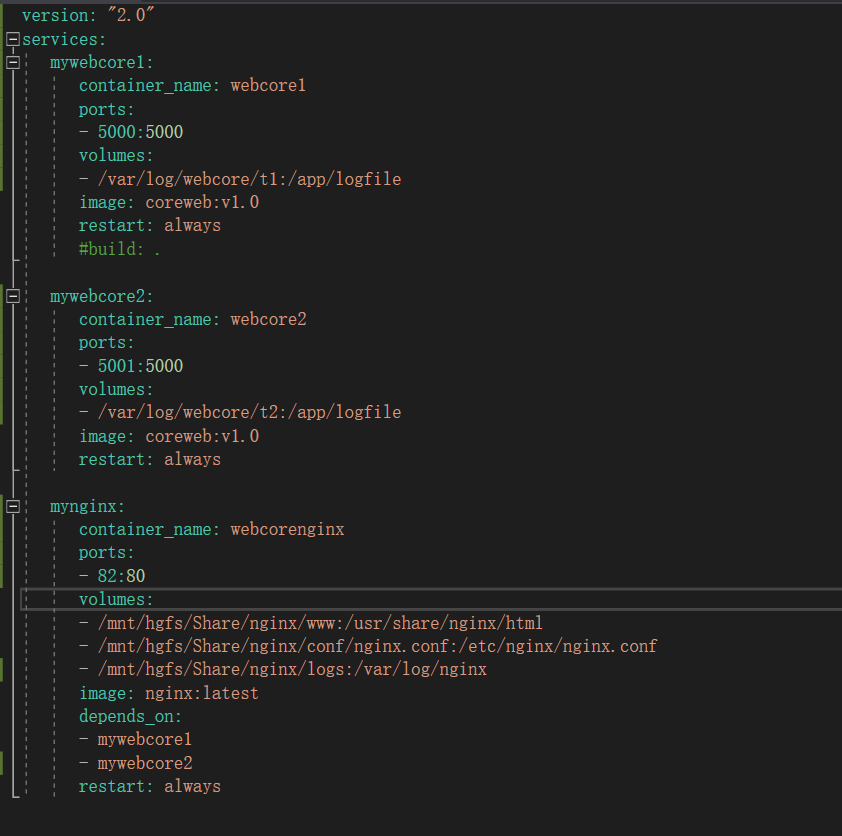


1. 运行编排

docker-compose up –d



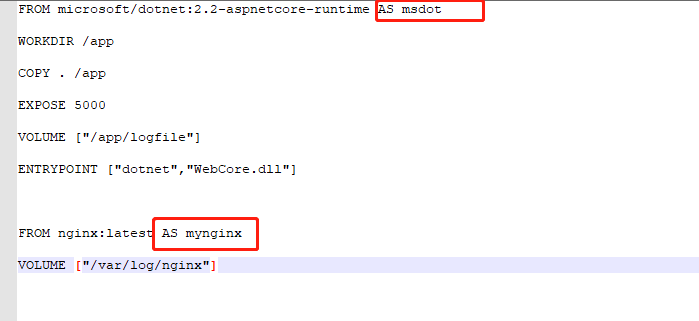
特别注意：使用ngxin和docker-compose，负载均衡的配置要注意：在nginx.conf配置docker-compose.yml中的服务名+容器中的原始端口（非映射后的）





Dockerfile构建多个镜像：

改写如：



C:\Users\tem\AppData\Local\Temp\1564566378(1).png

C:\Users\tem\AppData\Local\Temp\1564566398(1).png

