**[docker network基础](https://www.cnblogs.com/jsonhc/p/7823286.html)**

前面介绍了nginx与php两个容器间是如何进行通信的：

[root@docker ~]# docker run -d --name=php -v /www:/usr/local/nginx/html php

[root@docker ~]# docker run -d --name=nginx --link=php:php -v /www:/usr/local/nginx/html -p 81:80 nginx

[root@docker ~]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

58280fe851f9 nginx "/usr/local/nginx/..." 15 seconds ago Up 14 seconds 0.0.0.0:81->80/tcp nginx

9ea150c35587 php "/usr/local/php/sb..." 36 seconds ago Up 35 seconds 9000/tcp php

通过容器间的选项--link指定容器名称进行不同容器间的通信（--link container\_name或者将container\_name取一个别名）

现在使用另外一种方式替代--link来达到容器间的通信：docker network

查看local的网络信息：

[root@docker ~]# docker network ls

NETWORK ID NAME DRIVER SCOPE

5133ec415c3c bridge bridge local

f359ca4e2d39 host host local

8d68673c045c none null local

现在创建一个网络名为my\_net且driver为bridge的网络：（默认创建的就是bridge）

[复制代码](javascript:void(0);)

[root@docker ~]# docker network create my\_net

67e29f0e4a77c79144efc337a081a889188b5b8e289968f22be6e4ddd9b80610

[root@docker ~]# docker network ls

NETWORK ID NAME DRIVER SCOPE

5133ec415c3c bridge bridge local

f359ca4e2d39 host host local

67e29f0e4a77 my\_net bridge local

8d68673c045c none null local

[复制代码](javascript:void(0);)

利用--network启动容器提供服务：

[root@docker ~]# docker run -d --name=php --network my\_net --network-alias php -v /www:/usr/local/nginx/html php

6b493cbe8207dee4cb4d5945cfce305dba96914083bd7f46841b0b42376bcb99

[root@docker ~]# docker run -d --name=nginx --network my\_net --network-alias nginx -v /www:/usr/local/nginx/html -p 80:80 nginx

5ab220196b52bb768bef508433f0b920eecee70c3ee47880ebc5e2a74b5ee254

通过选项--network-alias将取名的my\_net起了一个别名

[root@docker ~]# docker exec -it nginx ping php

PING php (172.18.0.2) 56(84) bytes of data.

64 bytes from php.my\_net (172.18.0.2): icmp\_seq=1 ttl=64 time=0.079 ms

64 bytes from php.my\_net (172.18.0.2): icmp\_seq=2 ttl=64 time=0.090 ms

这里为啥能够ping通过php这个容器呢，因为这两个容器在同一个网络m'y\_net内，而nginx里面ping的php是容器名（这里管理的是容器级别）

由于能够ping通php，所以在nginx中配置的：

[复制代码](javascript:void(0);)

server {

listen 80;

root /usr/local/nginx/html;

index index.htm index.html index.php;

location ~ \.php$ {

root /usr/local/nginx/html;

fastcgi\_pass php:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

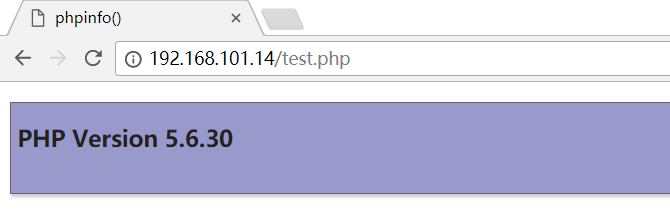
include fastcgi\_params;

}

}

[复制代码](javascript:void(0);)

所以配置文件中的php不会导致nginx启动失败，通过network的driver bridge实现了容器间的访问



上面在容器启动的时候使用的是选项--network，而在compose的配置文件中则是networks，现在通过配置文件来进行阐述该参数的作用：

[复制代码](javascript:void(0);)

[root@docker lnmp]# cat lnmp.yml

version: '3'

services:

nginx:

image: nginx

container\_name: lnmp-nginx

depends\_on:

- php

ports:

- "80:80"

networks:

- "net1"

volumes:

- "/www:/usr/local/nginx/html"

php:

image: php

container\_name: lnmp-php

expose:

- "9000"

networks:

- "net1"

volumes:

- "/www:/usr/local/nginx/html"

networks:

net1:

driver: bridge

[复制代码](javascript:void(0);)

由上述文件中可以知道networks定义了一个名称为net1的网络，由于networks是top-level（顶层级别，所以需要在顶层设置），而在创建的网络时候需要指定driver（单一网络使用bridge，swarm集群使用overlay），而且driver内容不能省略，在nginx和php两个service中使用了同一网络net1，现在将服务启动：

[复制代码](javascript:void(0);)

[root@docker lnmp]# docker-compose -f lnmp.yml up

Creating network "lnmp\_net1" with driver "bridge"

Creating lnmp-php ...

Creating lnmp-php ... done

Creating lnmp-nginx ...

Creating lnmp-nginx ... done

Attaching to lnmp-php, lnmp-nginx

[复制代码](javascript:void(0);)

可以看出在启动服务的时候创建了服务级别的网络lnmp\_net1

[复制代码](javascript:void(0);)

[root@docker lnmp]# docker network ls

NETWORK ID NAME DRIVER SCOPE

5133ec415c3c bridge bridge local

f359ca4e2d39 host host local

29d798852b52 lnmp\_net1 bridge local

67e29f0e4a77 my\_net bridge local

8d68673c045c none null local

[复制代码](javascript:void(0);)

服务nginx和php在网络lnmp\_net1实现了互联通信

[root@docker lnmp]# docker-compose -f lnmp.yml exec nginx ping php

PING php (172.19.0.2) 56(84) bytes of data.

64 bytes from lnmp-php.lnmp\_net1 (172.19.0.2): icmp\_seq=1 ttl=64 time=0.060 ms

在nginx服务中能够访问php的服务，在compose编排中级别的访问是对service级别的访问，所以在nginx服务配置文件中对应的php能够实现，不会导致nginx启动失败

及在compose的配置文件设置并对应的是service级别，当使用参数links也是配置的php服务，使得nginx服务能够ping通php服务并实现访问通信

[复制代码](javascript:void(0);)

[root@docker lnmp]# cat lnmp.yml

version: '3'

services:

nginx:

image: nginx

container\_name: lnmp-nginx

depends\_on:

- php

ports:

- "80:80"

links:

- php:php

volumes:

- "/www:/usr/local/nginx/html"

php:

image: php

container\_name: lnmp-php

expose:

- "9000"

volumes:

- "/www:/usr/local/nginx/html"

[复制代码](javascript:void(0);)

将networks参数改为了links参数，使得两个服务能够继续互联访问

[复制代码](javascript:void(0);)

[root@docker lnmp]# docker-compose -f lnmp.yml up

Creating network "lnmp\_default" with the default driver

Creating lnmp-php ...

Creating lnmp-php ... done

Creating lnmp-nginx ...

Creating lnmp-nginx ... done

Attaching to lnmp-php, lnmp-nginx

[复制代码](javascript:void(0);)

而且启动的时候创建的默认的网络

[root@docker lnmp]# docker-compose -f lnmp.yml exec nginx ping php

PING php (172.19.0.2) 56(84) bytes of data.

64 bytes from lnmp-php.lnmp\_default (172.19.0.2): icmp\_seq=1 ttl=64 time=0.091 ms

64 bytes from lnmp-php.lnmp\_default (172.19.0.2): icmp\_seq=2 ttl=64 time=0.107 ms

--links也是能够使得服务间进行访问的

现在通过案例讲解参数external\_links的作用：external\_links类似于links针对的也是service级别，但是external\_links对应的是外部的service而不是同一compose配置文件中的service

[复制代码](javascript:void(0);)

[root@docker lnmp]# cat lnmp.yml

version: '3'

services:

nginx:

image: nginx

container\_name: lnmp-nginx

depends\_on:

- php

ports:

- "80:80"

networks:

- "net1"

volumes:

- "/www:/usr/local/nginx/html"

external\_links:

- php1:php

php:

image: php

container\_name: lnmp-php

expose:

- "9000"

networks:

- "net1"

volumes:

- "/www:/usr/local/nginx/html"

networks:

net1:

driver: bridge

[复制代码](javascript:void(0);)

上面可以看见external\_links配置了php1的服务，并且将php1的服务也取别名为了php，然后编写另一个compose配置文件，并配置服务名为php1的服务：

[复制代码](javascript:void(0);)

[root@docker lnmp]# cat php1.yml

version: '3'

services:

php1:

image: php

container\_name: lnmp-php1

expose:

- "9000"

networks:

- "net1"

volumes:

- "/www:/usr/local/nginx/html"

networks:

net1:

driver: bridge

[复制代码](javascript:void(0);)

结合两个compose配置文件，可以知道有nginx、php、php1的服务，并且php1的服务也有php的别名，现在依次开启php1，和nginx与php：

[root@docker lnmp]# docker-compose -f php1.yml up

Creating network "lnmp\_net1" with driver "bridge"

Creating lnmp-php1 ...

Creating lnmp-php1 ... done

Attaching to lnmp-php1

[复制代码](javascript:void(0);)

[root@docker lnmp]# docker-compose -f lnmp.yml up --build

WARNING: Found orphan containers (lnmp-php1) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.

Creating lnmp-php ...

Creating lnmp-php ... done

Creating lnmp-nginx ...

Creating lnmp-nginx ... done

Attaching to lnmp-php, lnmp-nginx

[复制代码](javascript:void(0);)

查看启动的容器状态：

[root@docker www]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

a089cd261279 nginx "/usr/local/nginx/..." 25 seconds ago Up 25 seconds 0.0.0.0:80->80/tcp lnmp-nginx

4ff4bf7c1bf9 php "/usr/local/php/sb..." 25 seconds ago Up 25 seconds 9000/tcp lnmp-php

4e39cdf62bf1 php "/usr/local/php/sb..." 51 seconds ago Up 51 seconds 9000/tcp lnmp-php1

现在将nginx一起的php的这个服务停止掉，看下nginx服务是否将会被停止：

[root@docker lnmp]# docker-compose -f lnmp.yml stop php

Stopping lnmp-php ... done

而在nginx这里的输出如下：

lnmp-php exited with code 0

[root@docker www]# docker ps -a

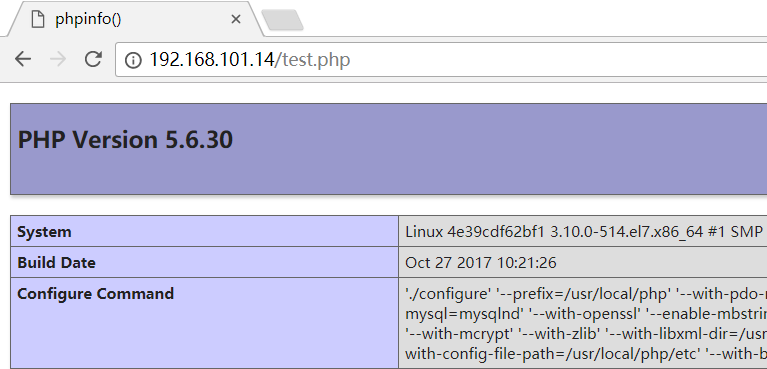
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

a089cd261279 nginx "/usr/local/nginx/..." 2 minutes ago Up 2 minutes 0.0.0.0:80->80/tcp lnmp-nginx

4ff4bf7c1bf9 php "/usr/local/php/sb..." 2 minutes ago Exited (0) 52 seconds ago lnmp-php

而php服务对应的容器lnmp-php也停止了，但是nginx的服务却没有停止，为什么呢？那是因为php1的服务并没有停止，而php1的服务也是别名php服务的，而external\_links就是实现这种情况的

使一个compose中的服务能与另一个compose中的服务能够互联通信，compose就是service级别的实现，配置文件中对应的都是service级别，而不是容器名



而在另一个compose中也必须定义与之相同的网络名称，也就是说配置external\_links时，两者服务间的网络名称必须一致，不然依旧不能进行互相访问

当使用了external\_links时，必须启动另一个compose的service，如果仅仅启动的是包含此函数的compose那么service将会启动报错

[复制代码](javascript:void(0);)

[root@docker lnmp]# cat lnmp.yml\_external\_links

version: '3'

services:

nginx:

image: nginx

container\_name: lnmp-nginx

depends\_on:

- php

ports:

- "80:80"

networks:

- "net1"

volumes:

- "/www:/usr/local/nginx/html"

external\_links:

- php1:php

php:

image: php

container\_name: lnmp-php

expose:

- "9000"

networks:

- "net1"

volumes:

- "/www:/usr/local/nginx/html"

networks:

net1:

driver: bridge

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

[root@docker lnmp]# docker-compose -f lnmp.yml up --build

Creating network "lnmp\_net1" with driver "bridge"

Creating lnmp-php ...

Creating lnmp-php ... done

Creating lnmp-nginx ...

Creating lnmp-nginx ... done

Attaching to lnmp-php, lnmp-nginx

lnmp-nginx | nginx: [emerg] host not found in upstream "php" in /usr/local/nginx/conf/vhost/www.conf:7

lnmp-nginx exited with code 1

[复制代码](javascript:void(0);)

参数external\_links连接的外部php服务并没有启动，所以nginx还是启动报错