1、端口映射实现访问容器

1.从外部访问容器应用

在启动容器的时候，如果不指定对应的参数，在容器外部是无法通过网络来访问容器内部的网络应用和服务的。

当容器中运行一些网络应用，要让外部访问这些应用时，可以通过-p或-P参数来指定端口映射。当使用-P(大写P)标记时，Docker会随机映射一个端口到内部容器开放的网络端口(端口范围在Linux系统使用的端口之外，一般都过万)：

```
[root@docker ~]
# docker run -d --name nginx_1 -P nginx:latest
f769af3e98478b27b87e008f3ad785e2055da4047442c4a8dcb8f621f810dbea
[root@docker ~]
# docker ps
CONTAINER ID    IMAGE       COMMAND         CREATED       STATUS       PORTS
NAMES
f769af3e9847    nginx:latest    "nginx -g 'daemon ..."3 seconds ago    Up 2 seconds
0.0.0.0:32768->80/tcp nginx_1
[root@docker ~]
#
```

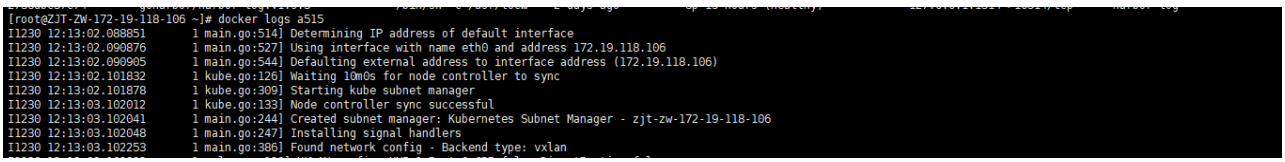
通过docker ps可以看到nginx_1容器的80端口被映射到本机的32768端口上。

这个地方有一个关键点就是用docker ps指令查询出来的PORTS的格式如：

0.0.0.0:32768->80/tcp 这表示前面是的本机的实际端口，后面是容器里面的端口

访问宿主主机的32768端口就可以访问容器内的应用程序提供的Web界面。

同样，可以通过docker logs命令查看应用信息：



-p(小写p)可以指定要映射的端口，并且在一个指定的端口上只可以绑定一个容器。支持的格式有：IP:HostPort:ContainerPort | IP::ContainerPort | HostPort:ContainerPort

2.映射所有接口地址

使用HostPort:ContainerPort格式将本地的5000端口映射到容器的5000端口：

[root@docker ~]# docker run -itd -p 5000:5000 --name nginx_2 nginx:latest

5bdca2bde33d7db72861399ca49e82f0d209d13289d20b181843da5b10e6f2d3

[root@docker ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

5bdca2bde33d nginx:latest "nginx -g 'daemon ..." 8 seconds ago Up 7 seconds 80/tcp, 0.0.0.0:5000->5000/tcp nginx_2

f769af3e9847 nginx:latest "nginx -g 'daemon ..." 15 minutes ago Up 15 minutes 0.0.0.0:32768->80/tcp nginx_1
[root@docker ~]#

此时默认会绑定本地所有接口上的所有地址。多次使用-p参数可以绑定多个端口：
[root@docker ~]# docker run -itd -p 3000:2700 -p 2389:8863 --name nginx_3 nginx:latest
65fbfbe9761eb5146501311016d681f210b1891ca5f5af62dc978ad6f2a22750
[root@docker ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
65fbfbe9761e nginx:latest "nginx -g 'daemon ..." 3 seconds ago Up 2 seconds 80/tcp, 0.0.0.0:3000->2700/tcp, 0.0.0.0:2389->8863/tcp nginx_3
5bdca2bde33d nginx:latest "nginx -g 'daemon ..." 2 minutes ago Up 2 minutes 80/tcp, 0.0.0.0:5000->5000/tcp nginx_2
f769af3e9847 nginx:latest "nginx -g 'daemon ..." 18 minutes ago Up 18 minutes 0.0.0.0:32768->80/tcp nginx_1
[root@docker ~]#

3.映射到指定地址的指定端口
可以使用IP:HostPort:ContainerPort格式指定映射使用一个特定地址：
[root@docker ~]# docker run -itd -p 10.0.0.31:89:8081 --name nginx_4 nginx:latest
16a476837222d413926053e1c8175c993b0495732073fbc6251dfd4696db8242
[root@docker ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
16a476837222 nginx:latest "nginx -g 'daemon ..." 4 seconds ago Up 3 seconds 80/tcp, 10.0.0.31:89->8081/tcp nginx_4
65fbfbe9761e nginx:latest "nginx -g 'daemon ..." 2 minutes ago Up 2 minutes 80/tcp, 0.0.0.0:3000->2700/tcp, 0.0.0.0:2389->8863/tcp nginx_3
5bdca2bde33d nginx:latest "nginx -g 'daemon ..." 5 minutes ago Up 5 minutes 80/tcp, 0.0.0.0:5000->5000/tcp nginx_2
f769af3e9847 nginx:latest "nginx -g 'daemon ..." 20 minutes ago Up 20 minutes 0.0.0.0:32768->80/tcp nginx_1
[root@docker ~]#

4.映射到指定地址的任意端口
使用IP::ContainerPort格式绑定本机的任意端口到容器的指定端口：
[root@docker ~]# docker run -itd -p 10.0.0.31::8082 --name nginx_5 nginx:latest
3436fd5fbdca6529c70c664f42edfd10d51edb0fb541b096b47c9b168887b2ca

```
[root@docker ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3436fd5fbdca nginx:latest "nginx -g 'daemon ..." 2 seconds ago Up 2 seconds 80/tcp,
10.0.0.31:32769->8082/tcp nginx_5
16a476837222 nginx:latest "nginx -g 'daemon ..." 2 minutes ago Up 2 minutes 80/tcp, 10.0.0.31:89-
>8081/tcp nginx_4
65fbfbe9761e nginx:latest "nginx -g 'daemon ..." 4 minutes ago Up 4 minutes 80/tcp, 0.0.0.0:3000-
>2700/tcp, 0.0.0.0:2389->8863/tcp nginx_3
5bdca2bde33d nginx:latest "nginx -g 'daemon ..." 7 minutes ago Up 7 minutes 80/tcp, 0.0.0.0:5000-
>5000/tcp nginx_2
f769af3e9847 nginx:latest "nginx -g 'daemon ..." 22 minutes ago Up 22 minutes 0.0.0.0:32768-
>80/tcp nginx_1
[root@docker ~]#
```

容器启动后，本机会随机自动分配一个未被占用的端口。

5.查看映射端口配置

使用docker port命令来查看当前映射的端口配置，也可以查看绑定的地址

```
[root@docker ~]# docker port nginx_1
80/tcp -> 0.0.0.0:32768
[root@docker ~]# docker port nginx_2
5000/tcp -> 0.0.0.0:5000
[root@docker ~]# docker port nginx_3
2700/tcp -> 0.0.0.0:3000
8863/tcp -> 0.0.0.0:2389
[root@docker ~]# docker port nginx_4
8081/tcp -> 10.0.0.31:89
[root@docker ~]# docker port nginx_5
8082/tcp -> 10.0.0.31:32769
[root@docker ~]#
```

注意：

容器有自己的内部网络和IP地址，使用docker inspect +容器ID可以获取容器的具体信息。

2、互联机制实现便捷访问

容器的互联是一种让多个容器中应用进行快速交互的方式，它会在源和接收容器之间建立连接关系，接收容器可以通过容器名快速访问到源容器，而不用指定具体的IP地址。

1.自定义容器命名

连接系统依据容器的名称来执行。因此，首先要给容器定义一个简单好记的名字。在容器创建的时候，系统会随机创建一个容器名，但是并没有特殊的意义也不便于记忆，所有自定义容器名有一下亮点好处：

1.自定义的命名比较好记，比如一个Web应用容器，就可以起名web，既方便记忆也方便理解容器的作用；

2.当要连接其他容器时，即使重启，也可以使用自定义的容器名，，比如web容器连接到db容器。

使用--name参数可以为容器自定义命名：

```
[root@docker ~]# docker run -itd --name centos_1 centos:latest
4d58a9f6f324185caf53dbe5eaee85f35e853842ffb037cf272c2a92cee89716
[root@docker ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4d58a9f6f324 centos:latest "/bin/bash" 6 seconds ago Up 5 seconds centos_1
3436fd5fbdca nginx:latest "nginx -g 'daemon ..." 11 minutes ago Up 11 minutes 80/tcp, 10.0.0.31:32769->8082/tcp nginx_5
16a476837222 nginx:latest "nginx -g 'daemon ..." 13 minutes ago Up 13 minutes 80/tcp, 10.0.0.31:89->8081/tcp nginx_4
65fbfbe9761e nginx:latest "nginx -g 'daemon ..." 15 minutes ago Up 15 minutes 80/tcp, 0.0.0.0:3000->2700/tcp, 0.0.0.0:2389->8863/tcp nginx_3
5bdca2bde33d nginx:latest "nginx -g 'daemon ..." 18 minutes ago Up 18 minutes 80/tcp, 0.0.0.0:5000->5000/tcp nginx_2
f769af3e9847 nginx:latest "nginx -g 'daemon ..." 34 minutes ago Up 34 minutes 0.0.0.0:32768->80/tcp nginx_1
[root@docker ~]#
```

通过docker ps或者docker ps -a可以查看到容器的自定义名字，利用docker inspect也可以获取到容器自定义名字：

```
[root@docker ~]# docker inspect -f "{{.Name}}" 4d58a9f6f324
/centos_1
[root@docker ~]#
```

注意：

容器的名称是唯一的。如果已经命名了一个web的容器，当再次使用web这个命名的时候会报错，如果一定要使用，需要先用docker rm删除之前创建的web容器。

在执行docker run的时候如果添加--rm参数，则容器终止后会立刻删除。--rm参数和-d参数不能同时使用。

2、容器互联

使用--link参数可以让容器之间安全地进行交互。

创建一个数据库容器：

[root@docker ~]# docker run -itd --name db --env MYSQL_ROOT_PASSWORD=example mariadb
b239b124946c99b7da63e00c22df802e9612fbe8bc636389205baf6c2f6963bd
[root@docker ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
b239b124946c mariadb "docker-entrypoint..." 3 seconds ago Up 2 seconds 3306/tcp db
[root@docker ~]#
创建一个web容器并将它连接到db容器：
[root@docker ~]# docker run -itd -P --name web --link db:db nginx:latest
42fa6662784010368b5e615d495e71920d85cc1bc089a5d181657514973ee90a
[root@docker ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
86ef0f632ffe nginx:latest "nginx -g 'daemon ..." 44 seconds ago Up 43 seconds 80/tcp web
b239b124946c mariadb "docker-entrypoint..." About a minute ago Up 59 seconds 3306/tcp db
[root@docker ~]#
此时web容器已经和db容器建立互联关系：--link参数的格式为：--link name:alias，其中name是要连接的容器名称，alias是这个连接的别名。
Docker相当于在两个互联的容器之间创建了一个虚拟通道，而不用映射它们的端口到宿主机上。在启动db容器的时候并没有使用-p或者-P参数，从而避免了暴露数据库服务端口到外部网络上。
Docker通过两种方式为容器公开连接信息：
1.更新环境变量；
2.更新/etc/hosts文件。
使用env命令来查看web容器的环境变量：
[root@docker ~]# docker run --rm --name web3 --link db:db nginx:latest env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=7258de738125
DB_PORT=tcp://172.17.0.2:3306
DB_PORT_3306_TCP=tcp://172.17.0.2:3306
DB_PORT_3306_TCP_ADDR=172.17.0.2
DB_PORT_3306_TCP_PORT=3306
DB_PORT_3306_TCP_PROTO=tcp
DB_NAME=/web3/db
DB_ENV_MYSQL_ROOT_PASSWORD=example
DB_ENV_GOSU_VERSION=1.10
DB_ENV_GPG_KEYS=199369E5404BD5FC7D2FE43BCBCB082A1BB943DB
430BDF5C56E7C94E848EE60C1C4CBDCDCD2EFD2A
4D1BB29D63D98E422B2113B19334A25F8507EFA5
DB_ENV_MARIADB_MAJOR=10.2

DB_ENV_MARIADB_VERSION=10.2.11+maria~jessie

NGINX_VERSION=1.13.7-1~stretch

NJS_VERSION=1.13.7.0.1.15-1~stretch

HOME=/root

[root@docker ~]#

其中DB_开头的环境变量是提供web容器连接db容器使用的，前缀采用大写的连接别名。

除了环境变量之外，Docker还添加host信息到父容器的/etc/hosts文件。

[root@docker ~]# docker run -it --rm --link db:db nginx:latest /bin/bash

root@16b8e6fde27f:/# cat /etc/hosts

172.17.0.2 db b239b124946c

172.17.0.5 16b8e6fde27f

root@16b8e6fde27f:/#

这里有两个hosts信息，第一个是db容器的IP和容器名+容器ID，第二个是web自己的IP和容器ID，web容器中hosts文件采用容器的ID作为主机名。

互联的容器之间是可以ping通的。