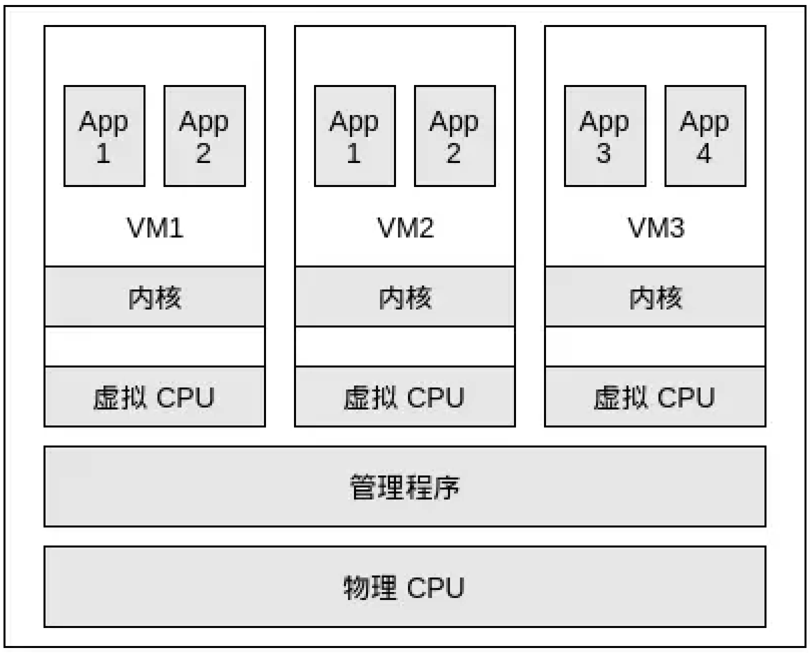
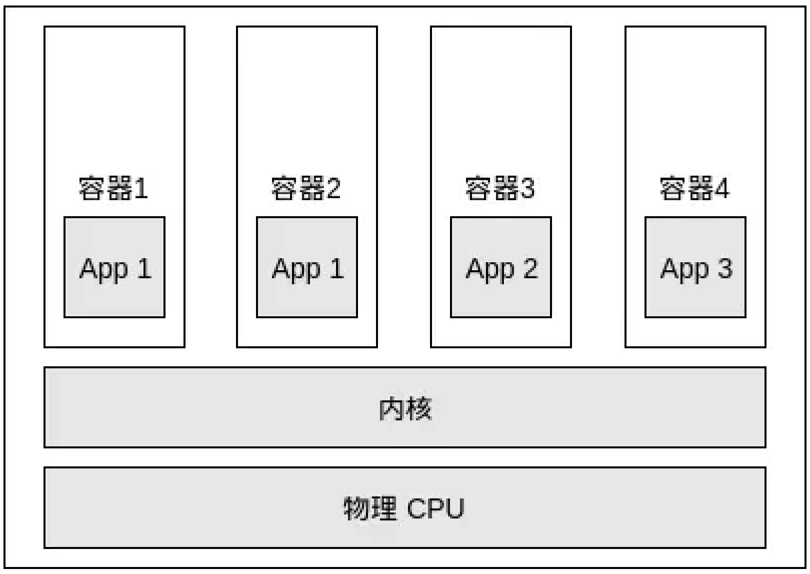
docker



虚拟化模式



容器化模式

docker 的三个基本概念：

1、镜像 Image

-- 实现了某个功能的代码模块（别人已经配置好的一个程序或者很多程序的一个环境）

-- 镜像 = 一个os + 一个程序或者多个程序 ----》人家已经做好的套餐盒饭

2、容器 Container

-- 将镜像里的代码运行起来的一个地方

-- 一个容器对应一个进程

3、仓库 Repository

-- 存放很多镜像的一个地方 --》(**时速云**)

-- 我们比较熟悉的一个仓库---》**yum**

操作使用docker：

*人生建议：*先升级yum--》yum update

安装docker：

yum install docker -y

启动docker：

```
[root@yun ~]# service docker start
Redirecting to /bin/systemctl start  docker.service
```

查看docker版本：

```
[root@yun ~]# service docker start
Redirecting to /bin/systemctl start  docker.service
[root@yun ~]# docker version
Client:
 Version:         1.13.1
 API version:     1.26
 Package version: docker-1.13.1-161.git64e9980.el7_8.x86_64
 Go version:      go1.10.3
 Git commit:      64e9980/1.13.1
 Built:           Tue Apr 28 14:43:01 2020
 OS/Arch:         linux/amd64

Server:
 Version:         1.13.1
 API version:     1.26 (minimum version 1.12)
 Package version: docker-1.13.1-161.git64e9980.el7_8.x86_64
 Go version:      go1.10.3
 Git commit:      64e9980/1.13.1
 Built:           Tue Apr 28 14:43:01 2020
 OS/Arch:         linux/amd64
 Experimental:    false
```

设置开机启动：

```
[root@yun ~]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
```

常用的docker命令：

docker version

docker images --- 查看docker镜像

docker search --- 查找docker镜像

- docker search --filter stars=200 nginx --- 过滤出stars在200以上的的nginx镜像

docker pull --- 下载镜像

docker logs --- 查看容器的日志

docker build --- 制作镜像

docker inspect --- 查看容器的底层信息

docker stop $(docker ps -a -q) ---- 停所有的docker服务

docker service updata --image nginx:new my_nginx

导入、导出镜像：

- docker save -o nginx_docker.tar docker.io/nginx --- 在本地的当前目录下创建一个名叫nginx_docker.tar的nginx镜像压缩文件。
- docker load < nginx_docker.tar --- 在其他机器上导入镜像。

- 也可以使用ftp，导出端--》安装vsftp,导出端--》安装lftp

docker ps -a --- 查看所有正在运行的容器

docker run -d -p 80:80 --name tzk_nginx docker.io/nginx

- -d：deamon起一个后台进程去运行
- -p：外面访问os的80端口，然后转发到容器里的80端口（端口映射）
- --name：给容器器命名
- 每一个运行的容器都有自己的一串唯一标识符

docker stop tzk_nginx --- 停掉运行的容器

docker restart tzk_nginx --- 重启容器

docker exec -it tzk_nginx /bin/bash --- 进入容器

exit --- 退出容器

docker rm --- 删除已有的容器

```
[root@yun ~]# docker search --filter stars=200 nginx
INDEX       NAME                                    DESCRIPTION

STARS     OFFICIAL   AUTOMATED
docker.io   docker.io/nginx                         Official build of Nginx.

  13263     [OK]
docker.io   docker.io/jwilder/nginx-proxy           Automated Nginx reverse
proxy fo                                                               r docker
c...    1813                [OK]
docker.io   docker.io/richarvey/nginx-php-fpm       Container running Nginx +
PHP-FP                                                               M capable
...    775                  [OK]
[root@yun ~]# docker pull docker.io/nginx
Using default tag: latest
Trying to pull repository docker.io/library/nginx ...
latest: Pulling from docker.io/library/nginx
afb6ec6fdc1c: Pull complete
b90c53a0b692: Pull complete
11fa52a0fdc0: Pull complete
Digest:
sha256:6fff55753e3b34e36e24e37039ee9eae1fe38a6420d8ae16ef37c92d1eb26699
Status: Downloaded newer image for docker.io/nginx:latest
[root@yun ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
    SIZE
docker.io/nginx     latest              9beeba249f3e        2 weeks ago
    127 MB

```

相关文件：

- config.v2.json：查看容器的信息
- resolv.conf：dns解析信息

- hostname：存放主机名
- hosts：存在域名解析信息

使用实例（redis、mysql）：

redis:

```
[root@mytest ~]# docker run -d -p 6379:6379 --name docker_redis docker.io/redis
c4739ca4bbaaa61df17707f642fe4e415d73d0a4a80396483ef6aeb939dc115c
[root@mytest ~]# docker stop docker_redis
docker_redis
[root@mytest ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                     PORTS               NAMES
1309c0808426        docker.io/redis     "docker-entrypoint..."   37 minutes ago      Exited (0) 6 seconds ago                       docker_redis
[root@mytest ~]# docker start docker_redis
docker_redis
[root@mytest ~]# docker exec -it docker_redis /bin/bash
root@1309c0808426:/data# redis-cli
127.0.0.1:6379>

```

mysql5.7:

-

可以通过改变端口，起多个容器使得一台机器(宿主机)上可以连接多个数据库！

```
[root@mytest ~]# docker pull docker.io/mysql:5.7

[root@mytest ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker.io/redis     latest              36304d3b4540        3 days ago          104 MB
docker.io/mysql     5.7                 a4fdfd462add        10 days ago         448 MB
[root@mytest ~]# docker run -d -p 3306:3306 --name my_mysql57 -e MYSQL_ROOT_PASSWORD='Tzkwan1314=' mysql:5.7
f57dec6fb9e531086afbe2696cffc37a2cad8903a202ff2b0d66c7b5efa4af1f
[root@mytest ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                               NAMES
f57dec6fb9e5        mysql:5.7           "docker-entrypoint..."   23 seconds ago      Up 22 seconds       0.0.0.0:3306->3306/tcp, 33060/tcp   my_mysql57

[root@mytest ~]# docker exec -it my_mysql57 /bin/bash
root@f57dec6fb9e5:/# mysql -uroot -pTzkwan1314=
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
```

```
Server version: 5.7.30 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql>
```

数据卷:

实现宿主机和容器进行数据共享平台

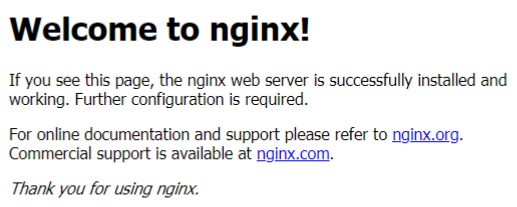
nginx-

将宿主机的/web/挂载到容器中的/usr/share/nginx/html

宿主机变容器变

```
[root@mytest web]# docker run -d -p 80:80 --name docker_nginx -v
/web:/usr/share/ng            inx/html nginx
60e4dd60173dd136612337c6de02fb2c0ff2cff202d770774777fd7dfe676e45
```

修改宿主机的index.html:

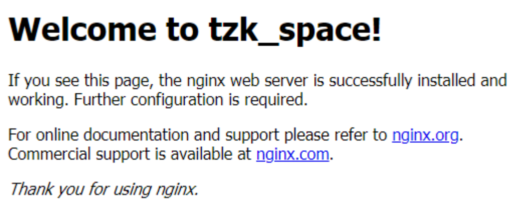

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

修改后:


Welcome to tzk_space!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

mysql-

1、先打一台容器，让宿主机的/mysql挂载进去获得文件数据：

```
[root@mytest mysql]# docker run -d -p 3307:3306 -v /mysql:/etc/mysql -v
/mysql_data                                       :/var/lib/mysql --name
my_mysql2 -e MYSQL_ROOT_PASSWORD='Tzkwan1314=' mysql:5.7
6d703a281972f65cef676b2ad8a1067b31a23ac16f0bba0841c82aa93a9f8833

root@ec3f26f40a26:/mnt# cp /etc/mysql . -r
root@ec3f26f40a26:/mnt# ls
mysql
root@ec3f26f40a26:/mnt# cp /var/lib/mysql mysql_data -r
root@ec3f26f40a26:/mnt# ls
mysql  mysql_data
```

2、查看宿主机上是否有数据过来：

```
[root@mytest mysql]# ls
mysql  mysql_data
```

3、再起一个新的容器，把这两个文件夹分别挂载到容器里的/etc/mysql /var/lib/mysql

```
[root@mytest mysql]# docker run -d -p 3308:3306 -v /mysql/mysql:/etc/mysql
-v /mysq                                      l/mysql_data:/var/lib/mysql -
-name my_mysql3 -e MYSQL_ROOT_PASSWORD='Tzkwan1314=' m
                      ysql:5.7
5d20a6b39a7f8d7d5076ebf7c8924e7782b176576edbd83d67dc2e6a5f74a314
[root@mytest mysql]# docker exec -it my_mysql3 bash
root@5d20a6b39a7f:/# mysql -uroot -pTzkwan1314=
mysql: [Warning] Using a password on the command line interface can be
insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.30 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

4、修改宿主机上的数据，查看容器重新连接mysql后的变化：

```
[root@mytest mysql]# ls
mysql  mysql_data
[root@mytest mysql]# cd mysql
[root@mytest mysql]# ls
conf.d  my.cnf  my.cnf.fallback  mysql.cnf  mysql.conf.d
```

```
[root@mytest mysql]# cd conf.d/
[root@mytest conf.d]# ls
docker.cnf  mysql.cnf  mysqldump.cnf
[root@mytest conf.d]# vim mysql.cnf

[mysql]
auto-rehash
prompt=\\u@\\d \\R:\\m mysql>

```

```
mysql> exit
Bye
root@5d20a6b39a7f:/# mysql -uroot -pTzkwan1314=
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.30 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

root@(none) 08:05 mysql>
```

---

docker的监控:

Prometheus: 普罗米修斯

---

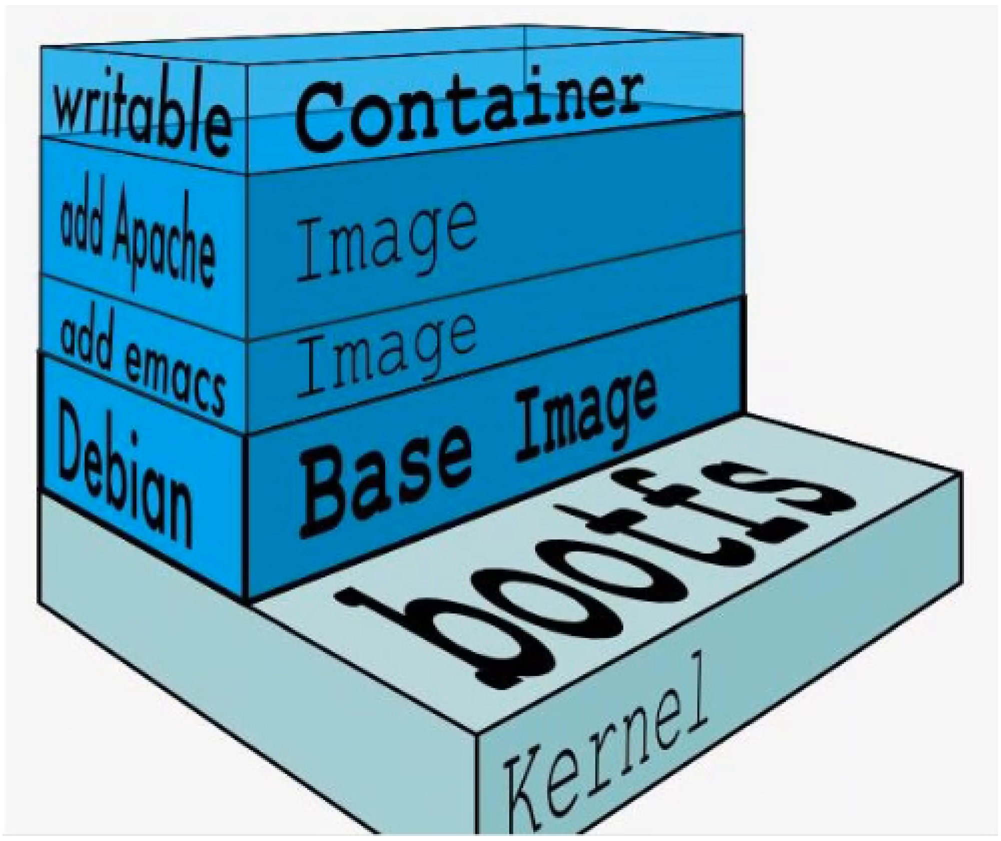
docker的网络类型:

1、默认是bright---》桥接

---

docker-镜像卷的制作:

https://www.cnblogs.com/panwenbin-logs/p/8007348.html



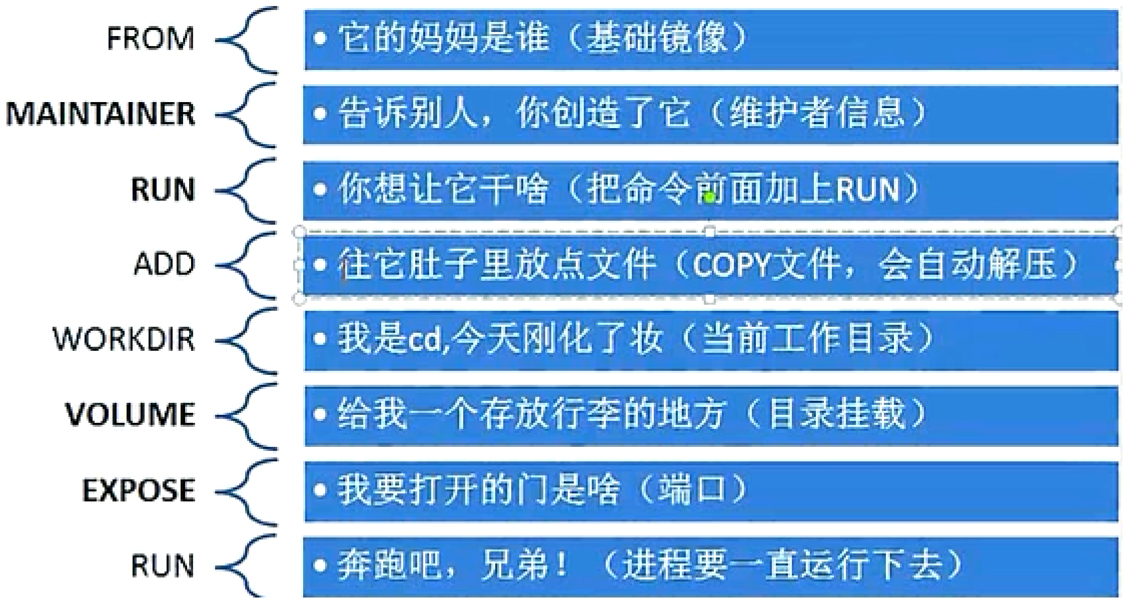
dockerfile：制作镜像文件的

什么是dockerfile：

Dockerfile是一个包含用于组合映像的命令的文本文档。可以使用在命令行中调用任何命令。 Docker通过读取`Dockerfile`中的指令自动生成映像。

`docker build`命令用于从Dockerfile构建映像。可以在`docker build`命令中使用`-f`标志指向文件系统中任何位置的Dockerfile。

-



FROM	• 它的妈妈是谁（基础镜像）
MAINTAINER	• 告诉别人，你创造了它（维护者信息）
RUN	• 你想让它干啥（把命令前面加上RUN）
ADD	• 往它肚子里放点文件（COPY文件，会自动解压）
WORKDIR	• 我是cd,今天刚化了妆（当前工作目录）
VOLUME	• 给我一个存放行李的地方（目录挂载）
EXPOSE	• 我要打开的门是啥（端口）
RUN	• 奔跑吧，兄弟！（进程要一直运行下去）

制作lnmp镜像

1、在一个文件夹下新建一个Dockerfile文件：

```
# 会安装一个centos最新版本的镜像
```

```
FROM centos
# 类似于说明制作这个镜像的作者--》tzk
maintainer dt_tzk
# 定义环境变量
env company daishuyun

env PATH /usr/sbin:$PATH
# 在容器中添加的操作
run yum install epel-release net-tools vim lsof -y \
    && yum install nginx -y \
    && yum install mariadb mariadb-server -y
# 定义一个工作目录--》/
workdir /
# 将宿主机的文件复制到容器里去
add flask/ /nginx/flask
add nginx.conf /backup
# 开放容器的端口：80
expose 80
# 定义容易受到SIGTERM信号时，停止运行
stopsignal SIGTERM
# 定义进入容器执行的第一条命令
# 不允许后台启动nginx
cmd ["nginx","-g","daemon off;"]
```

---

docker-集群管理--swarm

https://www.cnblogs.com/zhujingzhi/p/9792432.html

环境准备：

- 3台机器，一台做管理主机(172.16.101.219)，两台做节点机器(172.16.100.165\172.16.100.232)

1、修改hosts文件

```
[root@manager219 ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

172.16.100.165  node1
172.16.100.232  node2
172.16.101.219  manager219
```

2、使用scp /etc/hosts root@...:/etc/hosts发送到节点主机上

3、关闭三台机器上的防火墙

```
systemctl stop firewalld
systemctl disable firewalld
```

4、在三台机器上安装好docker以后，开始在管理主机上创建swarm并添加节点

```
[root@manager219 ~]# docker swarm init --advertise-addr 172.16.101.219
Swarm initialized: current node (gxtkdox9282kdhpvpdk65m27t) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-10gjcrbpygm18o9ba5gncppmkolwcc4456dzo9cvd1osyfm81l-0a3e3vnctvbx09qt1s0y47hjn 172.16.101.219:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

- 上面命令执行后，该机器自动加入到swarm集群。这个会创建一个集群token，获取全球唯一的token，作为集群唯一标识。后续将其他节点加入集群都会用到这个token值。
  其中，--advertise-addr参数表示其它swarm中的worker节点使用此ip地址与manager联系。命令的输出包含了其它节点如何加入集群的命令。
- 添加节点主机到swarm集群：在节点机器上输入

```
[root@node1 ~]# docker swarm join --token SWMTKN-1-10gjcrbpygm18o9ba5gncppmkolwcc4456dzo9cvd1osyfm81l-0a3e3vnctvbx09qt1s0y47hjn 172.16.101.219:2377
This node joined a swarm as a worker.

[root@node2 ~]# docker swarm join --token SWMTKN-1-10gjcrbpygm18o9ba5gncppmkolwcc4456dzo9cvd1osyfm81l-0a3e3vnctvbx09qt1s0y47hjn 172.16.101.219:2377
This node joined a swarm as a worker.
```

```
[root@manager219 ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
gxtkdox9282kdhpvpdk65m27t *   manager219          Ready               Active              Leader              18.06.3-ce
rho20sq7w9u92dme7nese6v5w     node1               Ready               Active                                  18.06.3-ce
mthxilwkknfqwgc9yw0ofjxsw     node2               Ready               Active                                  18.06.3-ce
```

docker node ls：只能在管理机器上运行

5、创建网络后再部署nginx服务：

```
docker network create -d overlay nginx_net
docker network ls|grep nginx_net
docker network inspect nginx_net
```

```
docker service create --replicas 1 --network nginx_net --name my_nginx -p 80:80 nginx
# 创建了一个具有一个副本的nginx服务，使用的事nginx镜像
# 查看正在运行服务的列表
docker service ls
# 查看任务被调度到哪个节点机器上了
docker service ps my_nginx
```

6、在swarm中动态扩容nginx服务：**scale**

```
docker service scale my_nginx=4
# 增加到4个nginx服务
```

7、模拟宕机其中一台节点机器：**改节点机器上运行的容器会调度到别的节点机器上！！！---》“高可用”**

```
[root@node139 ~]# systemctl stop docker
[root@manager43 ~]# docker node ls
ID                              HOSTNAME            STATUS
AVAILABILITY         MANAGER STATUS      ENGINE VERSION
ppk7q0bjond8a58xja7in1qid *    manager43           Ready
Active               Leader              18.06.0-ce
mums8azgbrffnecp3q8fz70pl      node139             Down
Active                                   18.06.1-ce
z3n36maf03yjg7odghikuv574      node188             Ready
Active                                   18.06.1-ce

然后过一会查询服务的状态列表
[root@manager43 ~]# docker service ps my_nginx
ID                  NAME                IMAGE               NODE
     DESIRED STATE       CURRENT STATE                ERROR
PORTS
yzonph0zu7km        my_nginx.1          nginx:latest        manager43
    Running             Running about an hour ago
wb1cpk9k22rl        my_nginx.2          nginx:latest        node188
    Running             Running about a minute ago
mlprstt9ds5x         \_ my_nginx.2      nginx:latest        node139
    Shutdown            Running 4 minutes ago
rhbj4bcr4t2c        my_nginx.3          nginx:latest        manager43
    Running             Running about a minute ago
y09lk90tdzdp         \_ my_nginx.3      nginx:latest        node139
    Shutdown            Running 4 minutes ago
clolfl3zlvj0        my_nginx.4          nginx:latest        node188
    Running             Running 6 minutes ago
```

8、在swarm中动态缩容：***实际运用到618、双十一有大量流量时，动态扩容和缩容***

```
docker service  scale my_nginx=1
```

9、如果想升级镜像，可以动态升级----》**很便捷**

```
docker service update --image nginx:new my_nginx
```

---

compose

-- **类似于ansible的playbook，python编写的，docker中的插件**

-- **用yaml语言来配置应用程序的服务，使用一个命令就可以根据配置创建并启动所有的服务**

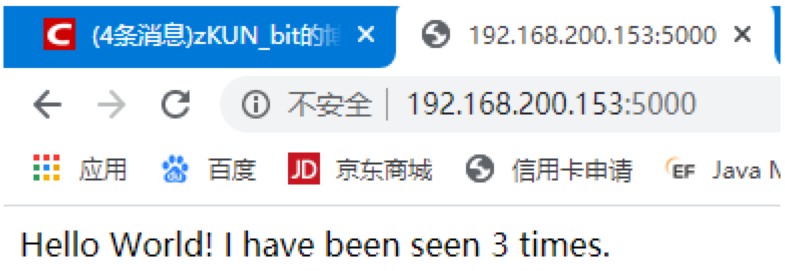
安装

```
curl -L "https://github.com/docker/compose/releases/download/1.26.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

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

ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

$ docker-compose --version
docker-compose version 1.26.0, build 1110ad01
```

上手

https://docs.docker.com/compose/gettingstarted/

gcc的下载有点久。。。



Hello World! I have been seen 3 times.

！！！遇到的问题

```
[root@yun ~]# docker run -d -p 6379:6379 --name tzk_redis docker.io/redis
2ec1101215785413c7e76f469ef98f54cde59ebeabba62402a5b16b454ac231c
/usr/bin/docker-current: Error response from daemon: oci runtime error: container_linux.go:235: starting container process caused "process_linux.go:258: applying cgroup configuration for process caused \"Cannot set property TasksAccounting, or unknown property.\""
```

问题原因： yum安装的docker版本低

解决方法：

- 删除下载的镜像 docker rmi <image_id>
- 升级yum：yum update

- yum remove docker
- yum install docker

```
[root@yun ~]# docker run -d -p 6379:6379 --name tzk_redis docker.io/redis
c4739ca4bbaaa61df17707f642fe4e415d73d0a4a80396483ef6aeb939dc115c
[root@yun ~]# lsof -i:6379
COMMAND     PID USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
docker-pr 7335 root    4u  IPv6 2126908      0t0  TCP *:6379 (LISTEN)
[root@yun ~]# redis-cli
127.0.0.1:6379> exit
```

！：

```
[root@mytest bin]# reids-cli
-bash: reids-cli: 未找到命令
```

运行redis的镜像后，要进入**交互模式的终端**

命令如下

```
docker exec -it myredis /bin/bash
```

执行redis-cli 就可以使用了

```
root@1309c0808426:/# redis-cli
127.0.0.1:6379> exit
```

面试经典问题：

1、容器里的数据是保存在哪里的？

/var/lib/docker/volumes

2、如果把容器停止或者docker服务停止，容器里的数据还会有吗？

有的

3、docker的底层隔离技术？

namespace　LXC

4、有哪些方法可以实现宿主机与容器之间数据共享？

数据卷、

Dockerfile：ADD、COPY

docker cp