Docker

# Docker安装

如果是在centos上安装docker，建议在7系列上安装。

安装依赖包

yum install -y yum-utils device-mapper-persistent-data lvm2

# 添加Docker软件包源

yum-config-manager \

--add-repo \

https://download.docker.com/linux/centos/docker-ce.repo

# 安装Docker CE

yum install docker-ce -y

# 启动

systemctl start docker

# 开机启动

systemctl enable docker

# 查看Docker信息

docker info

配置docker加速器

curl -sSL https://get.daocloud.io/daotools/set\_mirror.sh|sh -s http://bc437cce.m.daocloud.io

systemctl restart docker

# docker管理

## 创建一个nginx容器，并放入后台运行

docker container run -d nginx

## 列出当前正在运行的容器

docker container ls

指定容器名称、主机名和设置环境变量

docker container run -d --name nginx01 -e TEST=123 -h nginx01 nginx

## 进入容器查看

docker container exec -it nginx01 bash

root@nginx01:/# echo $TEST

123

root@nginx01:/# hostname

nginx01

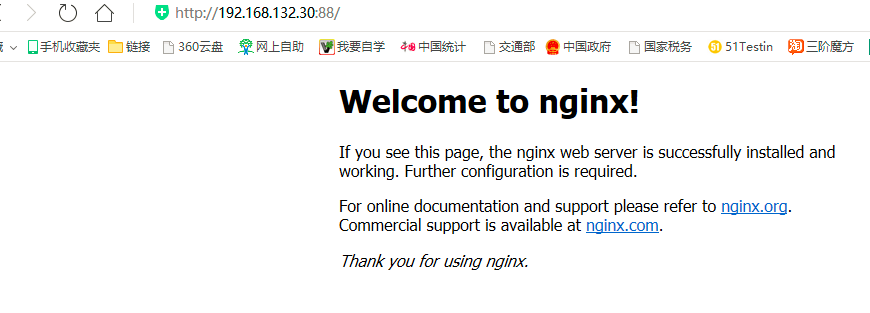
## 让用户访问容器

docker container run -d --name nginx02 -p 88:80 nginx

[root@salt-master ~]# netstat -tnlp|grep 88

tcp6 0 0 :::88 :::\* LISTEN 23224/docker-proxy

浏览器访问如下



## 容器资源管理

容器是密集型的，启动大量容器，如果不对容器进行资源限制，难免会因为某个容器占用大量资源，导致宿主机资源耗尽。资源限制常用选项。

执行docker run命令时能使用的和内存限制相关的所有选项如下

|  |  |
| --- | --- |
| 选项 | 描述 |
| -m,--memory | 内存限制，格式是数字加单位，单位可以为 b,k,m,g。最小为 4M |
| --memory-swap | 允许交换到磁盘中的内存量 |
| --memory-swappiness | 容器使用SWAP分区交换的百分比。（0-100，默认为-1 |
| --oom-kill-disable | 是否阻止 OOM killer 杀死容器，默认没设置 |
| --oom-score-adj | 容器被 OOM killer 杀死的优先级，范围是[-1000, 1000]，默认为 0 |
| --kernel-memory | 核心内存限制。格式同上，最小为 4M |
| --cpuset-cpus="" | 允许使用的 CPU 集，值可以为 0-3,0,1 |
| --cpus | 可以使用的cpu数量 |

限制容器内存使用

docker container run -d --name nginx03 --memory='100m' --memory-swap='100m' --oom-kill-disable nginx

查看当前容器内存限制及使用

[root@salt-master ~]# docker stats --no-stream nginx03

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

01fde633ae01 nginx03 0.00% 3.691MiB / 100MiB 3.69% 648B / 0B 2.1MB / 0B 2

cpu限额

允许容器最多使用50%的cpu

[root@salt-master ~]# docker container run -d --name nginx04 --cpus=".5" nginx

建议：设置memory再禁用oom killer，cpu使用不超过50%。

# 生产环境常用docker命令汇总

## docker info

显示 Docker 系统信息，包括镜像和容器数。

**docker version 显示docker版本信息**

## Docker pull

从docker仓库中下载镜像

如docker pull docker.io/nginx

## Docker run

基于docker run启动nginx镜像，并且启动到/bin/bash解释器；

docker run -itd docker.io/nginx /bin/bash

-i表示：interactive交互；

-t表示：tty终端；

d表示：daemon后台启动；

基于docker run启动nginx镜像，启动到/bin/bash解释器，同时映射本地80端口至容器80端口；

docker run -p 80:80 -itd docker.io/nginx /bin/bash

-p端口映射，第一个80宿主机监听端口，第二个80端口为容器监听;

浏览器输入宿主机IP+80端口，即可访问容器中的80端口所在服务

## Docker images

查看已下载的本地docker镜像列表

## Docker ps

查看当前正在运行中的容器，docker ps -a 查看当前Linux系统所有容器，包括运行和已经停止、其他容器（所有容器）

## Docker search

从docker仓库中搜索镜像

如docker search nginx docker search tomcat

## docker inspect

docker inspect 容器ID 查看容器详细信息

1. docker exec

进入docker容器操作相关命令

如在Docker中容器运行指令:df -h

[root@salt-master ~]# docker exec 1c1d2d958de9 df -h

Filesystem Size Used Avail Use% Mounted on

overlay 17G 7.5G 9.6G 44% /

tmpfs 64M 0 64M 0% /dev

tmpfs 493M 0 493M 0% /sys/fs/cgroup

/dev/mapper/cl-root 17G 7.5G 9.6G 44% /etc/hosts

shm 64M 0 64M 0% /dev/shm

tmpfs 493M 0 493M 0% /proc/acpi

tmpfs 493M 0 493M 0% /proc/scsi

tmpfs 493M 0 493M 0% /sys/firmware

docker exec -it 1c1d2d958de9 /bin/bash 进入容器进行交互

## docker kill

kill掉正在运行中的某个容器

docker kill 1c1d2d958de9

## docker rm

删除某个指定的已经停止Docker容器，加上-f参数表示强制删除某个指定的正在运行中的Docker容器

## docker rmi

从Docker images列表中删除某个镜像

还有其他一些命令，如下：

|  |  |  |
| --- | --- | --- |
| docker | push | 推送指定镜像或者库镜像至docker源服务器 |
| docker | history | 展示一个镜像形成历史 |
| docker | attach | 当前shell下attach连接指定运行镜像 |
| docker | build | 通过Dockerfile定制镜像 |
| docker | commit | 提交当前容器为新的镜像 |
| docker | cp | 从容器中拷贝指定文件或者目录到宿主机中 |
| docker | create | 创建一个新的容器，同run，但不启动容器 |
| docker | diff | 查看docker容器变化 |
| docker | events | 从docker服务获取容器实时事件 |
| docker | export | 导出容器的内容流作为一个tar归档文件[对应import] |
| docker | import | 从tar包中的内容创建一个新的文件系统映像[对应export] |
| docker | load | 从一个tar包中加载一个镜像[对应save] |
| docker | login | 注册或者登陆一个docker源服务器 |
| docker | logout | Dockerregistry退出 |
| docker | logs | 输出当前容器日志信息 |
| docker | port | 查看映射端口对应的容器内部源端口 |
| docker | pause | 暂停容器 |
| docker | restart | 重启运行的容器 |
| docker | save | 保存一个镜像为一个tar包[对应load] |
| docker | tag | 给源中镜像打标签 |
| docker | top | 查看容器中运行的进程信息 |
| docker | unpause | 取消暂停容器 |
| docker | wait | 截取容器停止时的退出状态值 |