努力哥-运维自动化

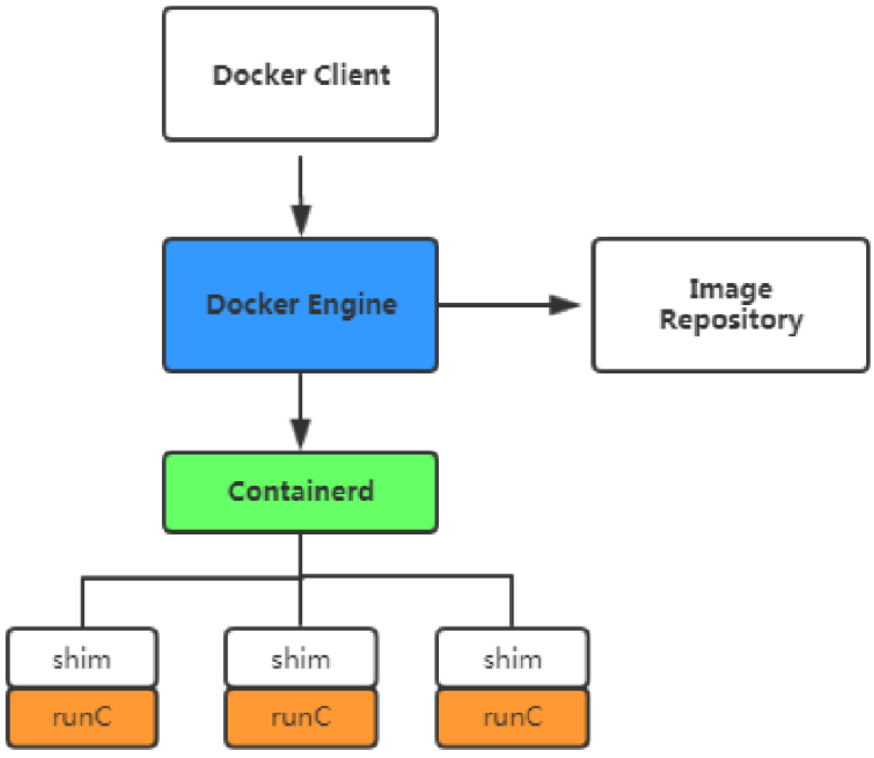
docker命令详细讲解（阿良）

一、docker详解

1.1、Docker的介绍

Docker是一个开源的应用容器引擎，使用Go语言开发，基于Linux内核的cgroup，namespace，Union FS等技术，对应用进程进行封装隔离，并且独立于宿主机与其他进程，这种运行时封装的状态称为容器。
Docker早起版本实现是基于LXC，并进一步对其封装，包括文件系统、网络互联、镜像管理等方面，极大简化了容器管理。从0.7版本以后开始去除LXC，转为自行研发的libcontainer，从1.11版本开始，进一步演进为使用runC和containerd。
Docker理念是将应用及依赖包打包到一个可移植的容器中，可发布到任意Linux发行版Docker引擎上。使用沙箱机制运行程序，程序之间相互隔离。

1.2、docker的体系架构



Containerd：是一个简单的守护进程，使用runC管理容器。向Docker Engine提供接口。
Shim：只负责管理一个容器。
runC：是一个轻量级的工具，只用来运行容器。

公告

昵称： 努力哥
园龄： 2年10个月
粉丝： 480
关注： 43
+加关注

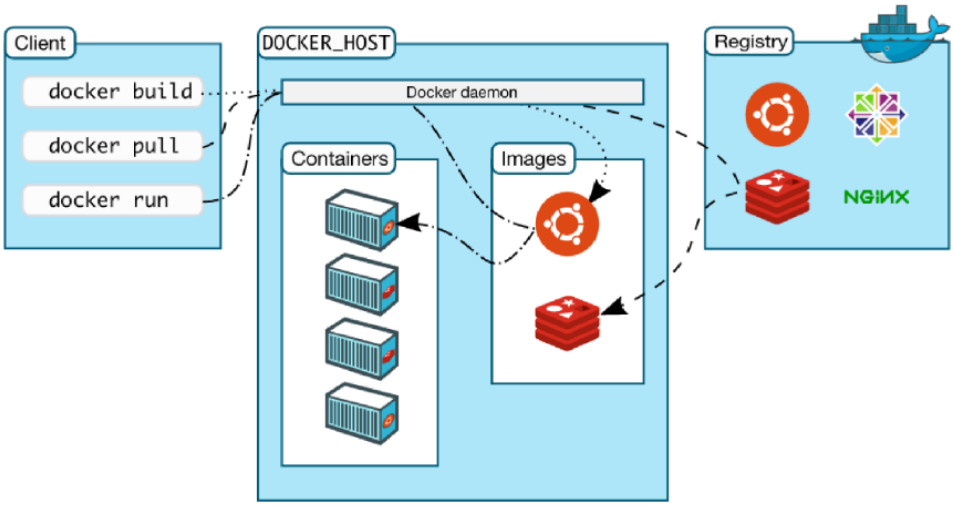
< 2019年9

日	一	二	三
1	2	3	4
8	9	10	11
15	16	17	18
22	23	24	25
29	30	1	2
6	7	8	9

搜索

我的标签

Python基础-模块(2)
python基础-内置函数详
python基础-生
python基础-实现购物车



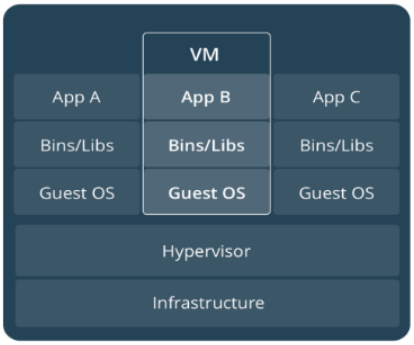
1.4、docker的内部组件

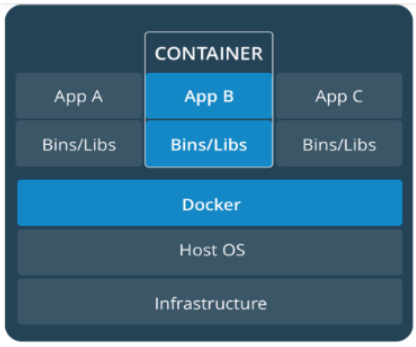
1. Namespaces
命名空间，Linux内核提供的一种对进程资源隔离的机制，例如进程、网络、挂载点等资源。

2. CGroups
控制组，Linux内核提供的一种限制进程资源的机制；例如CPU、内存等资源。

3. UnionFS
联合文件系统，支持将不同位置的目录挂载到同一虚拟文件系统，形成一种分层的模型。

1.5、虚拟机与容器区别





以 KVM 举例，与 Docker 对比

启动时间
Docker秒级启动，KVM分钟级启动。

轻量级
容器镜像大小通常以M为单位，虚拟机以G为单位。
容器资源占用小，要比虚拟机部署更快速。

性能
容器共享宿主机内核，系统级虚拟化，占用资源少，没有Hypervisor层开销，容器性能基本接近物理机；
虚拟机需要Hypervisor层支持，虚拟化一些设备，具有完整的GuestOS，虚拟化开销大，因而降低性能，没有容器性能好。

安全性
由于共享宿主机内核，只是进程级隔离，因此隔离性和稳定性不如虚拟机，容器具有一定权限访问宿主机内核，存在一定安全隐患。

使用 要求
KVM基于硬件的完全虚拟化，需要硬件CPU虚拟化技术支持；
容器共享宿主机内核，可运行在主流的Linux发行版，不用考虑CPU是否支持虚拟化技术。

1.6、docker 的应用场景

场景一：节省项目环境部署时间
1. 单项目打包
2. 整套项目打包
3. 新开源技术试用
场景二：环境一致性

python基础-实现用户密
Python基础-数据类型(1
python基础-通过Pytho
器(1)
Python基础-文件操作(1
python基础-文件读写&
python基础-修改haproc
更多

随笔分类
AI语音识别(1)
ansible(2)
Cobbler(2)
css(1)
Django框架(27)
docker(2)
ELK(1)
fedora(4)
Git(9)
gitlab
go语言(172)
Harbor(1)
html(1)
JavaScript (2)
Jenkins
jQuery(2)
kvm(3)
linux高级(9)

场景三：持续集成
场景四：微服务
场景五：弹性伸缩

参看： https://blog.51cto.com/lizhenliang/1978081

1.7、在Centos7.x安装docker

CentOS7

```
# 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 http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
# 更新yum包索引
yum makecache fast
# 安装Docker CE
yum install docker-ce -y
# 启动
systemctl start docker
# 卸载
yum remove docker-ce
rm -rf /var/lib/docker
```

官方安装文档：
https://docs.docker.com/engine/installation/linux/docker-ce/centos/#docker-ee-customers

1.8、镜像加速

什么是镜像?
简单说，Docker镜像是一个不包含Linux内核而又精简的Linux操作系统。

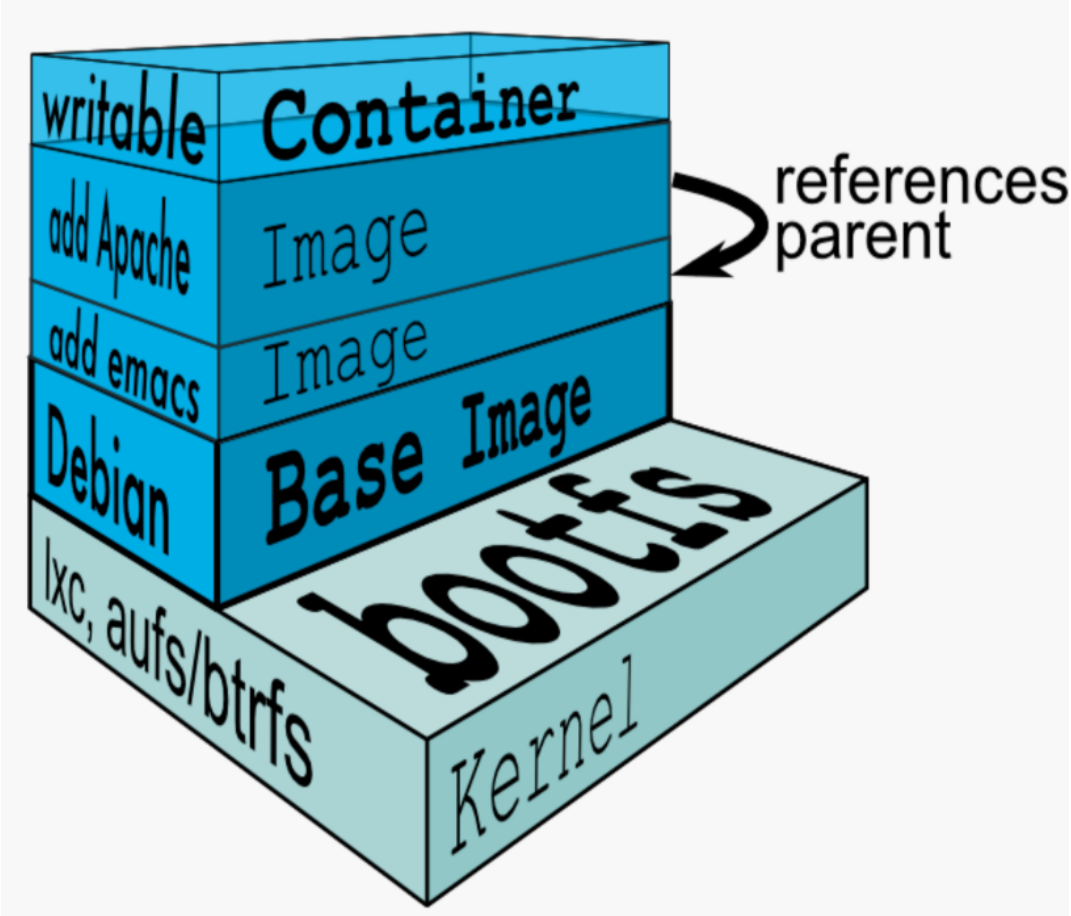
镜像从哪里来?
Docker Hub是由Docker公司负责维护的公共注册中心，包含大量的容器镜像，Docker工具默认从这个公共镜像库下载镜像。
https://hub.docker.com/explore
默认是国外的源，下载会慢，建议配置国内镜像仓库：

```
# vi /etc/docker/daemon.json
{
"registry-mirrors": [ "https://registry.docker-cn.com"]
}
```

重启一下：

```
systemctl restart docker
```

1.9、镜像与容器的关系



镜像不是一个单一的文件，而是有多层构成。我们可以通过docker history <ID/NAME> 查看镜像中各层内容及大小，每层对应着Dockerfile中的一条指令。Docker镜像默认存储在/var/lib/docker/<storage-driver>中。

容器其实是在镜像的最上面加了一层读写层，在运行容器里做的任何文件改动，都会写到这个读写层。如果容器删除了，最上面的读写层也就删除了，改动也就丢失了。

Docker使用存储驱动管理镜像每层内容及可读写层的容器层。

```
[root@node01 ~]# cd /var/lib/docker
[root@node01 docker]# ls
builder  containerd  containers  image  network  overlay2  plugins  runtimes  swarm  tmp  trust  volumes
[root@node01 docker]#
[root@node01 docker]#
[root@node01 docker]# ll
总用量 0
drwx------ 2 root root  24 4月  17 16:39 builder
drwx--x--x 3 root root  20 4月  17 16:39 containerd
drwx------ 2 root root   6 4月  17 16:39 containers
drwx------ 3 root root  22 4月  17 16:39 image
drwxr-x--- 3 root root  19 4月  17 16:39 network
drwx------ 6 root root 256 4月  17 16:57 overlay2
drwx------ 4 root root  32 4月  17 16:39 plugins
drwx------ 2 root root   6 4月  17 16:55 runtimes
drwx------ 2 root root   6 4月  17 16:39 swarm
drwx------ 2 root root   6 4月  17 16:57 tmp
drwx------ 2 root root   6 4月  17 16:39 trust
drwx------ 2 root root  25 4月  17 16:39 volumes
```

2.0、存储驱动

Linux distribution	Recommended storage drivers
Docker CE on Ubuntu	aufs , devicemapper , overlay2 (Ubuntu 14.04.4 or later, 16.04 or later), overlay , zfs , vfs
Docker CE on Debian	aufs , devicemapper , overlay2 (Debian Stretch), overlay , vfs
Docker CE on CentOS	devicemapper , vfs
Docker CE on Fedora	devicemapper , overlay2 (Fedora 26 or later, experimental), overlay (experimental), vfs

Storage driver	Supported backing filesystems
overlay , overlay2	ext4 , xfs
aufs	ext4 , xfs
devicemapper	direct-lvm
btrfs	btrfs
zfs	zfs

2.1、镜像命令

指令	描述
ls	列出镜像
build	构建镜像来自Dockerfile
history	查看镜像历史
inspect	显示一个或多个镜像详细信息
pull	从镜像仓库拉取镜像
push	推送一个镜像到镜像仓库
rm	移除一个或多个镜像
prune	移除未使用的镜像。没有被标记或被任何容器引用的。
tag	创建一个引用源镜像标记目标镜像
export	导出容器文件系统到tar归档文件
import	导入容器文件系统tar归档文件创建镜像
save	保存一个或多个镜像到一个tar归档文件
load	加载镜像来自tar归档或标准输入

二、docker常用命令

2.1、查看版本

```
[root@ansible-server ~]# docker --version
Docker version 18.09.6, build 481bc77156
```

2.2、查看帮助

```
[root@ansible-server ~]# docker --help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/root/.docker")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/root/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/root/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/root/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit
```

2017年6月(15)
2017年5月(31)
2017年4月(13)
2017年3月(35)
2017年2月(24)
2017年1月(6)
2016年12月(26)
2016年11月(49)

文章分类
harbor(2)
jenkins(2)
Ansible(1)
apache(1)
Ceph
docker(33)
ELK(5)
Gitlab(5)
Kafka(1)
Kubernetes & k8s(10
kvm(2)
linux(19)
Mysql(9)
nginx(4)
Openstack
python
Redis(4)

2.3、查看镜像分层

```
[root@ansible-server ~]# docker image history nginx:latest
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
53f3fd8007f7        2 weeks ago         /bin/sh -c #(nop)  CMD ["nginx" "-g" "daemon…   0B
<missing>           2 weeks ago         /bin/sh -c #(nop)  STOPSIGNAL SIGTERM           0B
<missing>           2 weeks ago         /bin/sh -c #(nop)  EXPOSE 80                    0B
<missing>           2 weeks ago         /bin/sh -c ln -sf /dev/stdout /var/log/nginx…   22B
<missing>           2 weeks ago         /bin/sh -c set -x  && apt-get update  && apt…   54.1MB
<missing>           2 weeks ago         /bin/sh -c #(nop)  ENV NJS_VERSION=1.15.12.0…   0B
<missing>           2 weeks ago         /bin/sh -c #(nop)  ENV NGINX_VERSION=1.15.12…   0B
<missing>           2 weeks ago         /bin/sh -c #(nop)  LABEL maintainer=NGINX Do…   0B
<missing>           2 weeks ago         /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>           2 weeks ago         /bin/sh -c #(nop) ADD file:fcb9328ea4c115670…   55.3MB
```

2.4、查看镜像的详细信息

查看镜像的详细信息

```
[root@ansible-server ~]# docker image inspect nginx
[
    {
        "Id": "sha256:53f3fd8007f76bd23bf663ad5f5009c8941f63828ae458cef584b5f85dc0a7bf",
        "RepoTags": [
            "nginx:latest"
        ],
        "RepoDigests": [
省略部分......
```

2.5、下载镜像

```
[root@ansible-server ~]# docker image pull nginx:1.11
1.11: Pulling from library/nginx
6d827a3ef358: Pull complete
f8f2e0556751: Pull complete
5c9972dca3fd: Pull complete
451b9524cb06: Pull complete
Digest: sha256:e6693c20186f837fc393390135d8a598a96a833917917789d63766cab6c59582
Status: Downloaded newer image for nginx:1.11
```

2.6、删除镜像

查看并删除镜像

```
[root@ansible-server ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              53f3fd8007f7        2 weeks ago         109MB
nginx               1.11                5766334bdaa0        2 years ago         183MB
<br>[root@ansible-server ~]# docker image rm nginx:1.11
Untagged: nginx:1.11
Untagged: nginx@sha256:e6693c20186f837fc393390135d8a598a96a833917917789d63766cab6c59582
Deleted: sha256:5766334bdaa0bc37f1f0c02cb94c351f9b076bcffa042d6ce811b0fd9bc31f3b
Deleted: sha256:1fcf2d3addf02c3b6add24c7b0993038f7e3eee616b10e671e25440e03bc7697
Deleted: sha256:51c56cdbb9306c4d6f2da2b780924f3b926bd13d15a4f6693a5175690e288436
Deleted: sha256:ec9a826666cfa5df0471f716145da63294019c09a5f2e31613122b57df8f7ce0
Deleted: sha256:5d6cbe0dbcf9a675e86aa0fbedf7ed8756d557c7468d6a7c64bde7fa9e029636
```

2.7、给镜像打tag

```
#给镜像打tag，再查看
[root@ansible-server ~]# docker tag nginx:1.11 nginx:v1

[root@ansible-server ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              53f3fd8007f7        2 weeks ago         109MB
nginx               1.11                5766334bdaa0        2 years ago         183MB
nginx               v1                  5766334bdaa0        2 years ago         183MB
```

2.8、导出镜像

```
[root@ansible-server ~]# docker image save nginx:1.11 >nginx1.11.tar
[root@ansible-server ~]# du -sh nginx1.11.tar
182M    nginx1.11.tar
```

2.9、导入镜像

```
#删除这个已存在的镜像
[root@ansible-server ~]# docker rmi nginx:1.11
Untagged: nginx:1.11
```

Saltstack(3)
ss & vpn(1)
tengine(1)
ubuntu
Zabbix(1)
ZooKeeper(2)
公司项目(2)

最新评论

1. Re:Django安装和配
说的很详细耶

2. Re:go语言之进阶篇F
好像这个源码编译的得至
报错

3. Re:go语言之进阶篇
好像这个源码编译的不能

4. Re:更改Windows用
s）默认位置到其它盘
适用于win10系统吗?
--

5. Re:remmina rdp远
这意思是windows上也可
a,然后远程连接ubuntu
关教程?

阅读排行榜

1. Linux下如何查看系统
时间以及安装时间(6316

2. 运维工程师
7)

```
#再导入镜像
[root@ansible-server ~]# docker load <nginx1.11.tar
Loaded image: nginx:1.11

#查看镜像
[root@ansible-server ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              53f3fd8007f7        2 weeks ago         109MB
nginx               1.11                5766334bdaa0        2 years ago         183MB
nginx               v1                  5766334bdaa0        2 years ago         183MB
```

3.0、运行一个容器

```
[root@ansible-server ~]# docker run -itd nginx
b8ecef224d29f0eaece24c9406e88207491443ab6beb053eb560dce2171b8b4a

#查看容器
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
b8ecef224d29        nginx               "nginx -g 'daemon of…"   9 seconds ago       Up 6 seconds        80/tcp
affectionate_feistel
```

3.1、导出一个正在运行的容器（备注：导出后就变成了一个镜像文件）

```
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
b8ecef224d29        nginx               "nginx -g 'daemon of…"   2 minutes ago       Up 2 minutes        80/tcp
affectionate_feistel

[root@ansible-server ~]# docker export b8ecef224d29 >nginx.tar

 [root@ansible-server ~]# du -sh nginx.tar
107M    nginx.tar
```

3.2、导入镜像

```
[root@ansible-server ~]# docker image import nginx.tar nginx:self
sha256:bbf50008d2bfb21486bb723cb9779ac04e854ee7e8176529d433941527a10fb9

 [root@ansible-server ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               self                594fc0cf36b0        58 seconds ago      108MB
nginx               latest              53f3fd8007f7        2 weeks ago         109MB
nginx               1.11                5766334bdaa0        2 years ago         183MB
nginx               v1                  5766334bdaa0        2 years ago         183MB
```

三、容器管理

3.1、查看容器命令

```
[root@ansible-server ~]# docker container --help

Usage:  docker container COMMAND

Manage containers

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  inspect     Display detailed information on one or more containers
  kill        Kill one or more running containers
  logs        Fetch the logs of a container
  ls          List containers
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  prune       Remove all stopped containers
  rename      Rename a container
  restart     Restart one or more containers
```

3. 运维监控系统之Oper
3)

4. MySQL各版本的区别

5. Python基础-字符串
式_format方式(40419

评论排行榜

1. django ajax增 删 改

2. 运维监控系统之Oper

3. CMDB开发(4)

4. tornado框架介绍(3)

5. 搭建企业内部yum仓
ntos7+epel源)(3)

推荐排行榜

1. python基础-文件读
(4)

2. Python并发编程-Ra
(4)

3. Mongodb副本集＋分
(4)

4. Mongodb主从复制
集群梳理(4)

5. python网络编程-soc

```
  rm          Remove one or more containers
  run         Run a command in a new container
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes

Run 'docker container COMMAND --help' for more information on a command.
```

3.2、创建容器常用选项

指令	描述	资源限制指令	描述
-i, --interactive	交互式	-m, --memory	容器可以使用的最大内存量
-t, --tty	分配一个伪终端	--memory-swap	允许交换到磁盘的内存量
-d, --detach	运行容器到后台	--memory-swappiness=<0-100>	容器使用SWAP分区交换的百分比（0-100，默认为-1）
-a, --attach list	附加到运行的容器	--memory-reservation	内存软限制，Docker检测主机容器争用或内存不足时所激活的软限制，使用此选项，值必须设置低于--memory，以使其优先
--dns list	设置DNS服务器	--oom-kill-disable	当宿主机内存不足时，内核会杀死容器中的进程。建议设置了-memory选项再禁用OOM。如果没有设置，主机可能会耗尽内存
-e, --env list	设置环境变量	--cpus	限制容器可以使用多少可用的CPU资源
--env-file list	从文件读取环境变量	--cpuset-cpus	限制容器可以使用特定的CPU
-p, --publish list	发布容器端口到主机	--cpu-shares	此值设置为大于或小于默认1024值，以增加或减少容器的权重，并使其可以访问主机CPU周期的更大或更小比例
-P, --publish-all	发布容器所有EXPOSE的端口到宿主机随机端口		
-h, --hostname string	设置容器主机名		
--ip string	指定容器IP，只能用于自定义网络		
--link list	添加连接到另一个容器		
--network	连接容器到一个网络		
--mount mount	挂载宿主机分区到容器		
-v, --volume list	挂载宿主机目录到容器		
--restart string	容器退出时重启策略，默认no [always|on-failure]		
--add-host list	添加其他主机到容器中/etc/hosts		

3.3、创建一个容器

```
#创建一个重命名为bs的容器
[root@ansible-server ~]# docker container run -itd --name bs busybox
27080338dabb3d76d3a5864999e2085240d3a6e9c7ef201bd91f9d18c0167969

#查看容器
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
27080338dabb        busybox             "sh"                     49 seconds ago      Up 47
seconds                                       bs
d5ee27264bd3        alpine              "/bin/sh"                3 minutes ago       Exited (0) About a minute
ago                       focus
b8ecef224d29        nginx               "nginx -g 'daemon of…"   45 minutes ago      Up 45 minutes
80/tcp              affec

#进入容器中
[root@ansible-server ~]# docker container attach bs
/ # ls
bin   dev   etc   home  proc  root  sys   tmp   usr   var
/ # ps -ef
PID   USER     TIME  COMMAND
    1 root      0:00 sh
    8 root      0:00 ps -ef

/ # ifconfig
eth0 Link encap:Ethernet HWaddr 02:42:AC:11:00:03
inet addr:172.17.0.3 Bcast:172.17.255.255 Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:8 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:648 (648.0 B) TX bytes:0 (0.0 B)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

/ # echo "123" >>/etc/hosts
```

```
/ # tail -1 /etc/hosts
123

/ # exit  #退出容器，同时终端也会并闭。
```

3.4、进入容器命令

```
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
27080338dabb        busybox             "sh"                     15 minutes ago      Up 4
minutes                                    bs

[root@ansible-server ~]# docker exec -it bs sh
/ # ls
bin   dev   etc   home  proc  root  sys   tmp   usr   var
/ #
```

3.5、运行容器，映射端口80到8088上面。

```
#运行容器，映射端口80到8088上面。
[root@ansible-server ~]# docker container run -itd -p 8080:80 --name nginx02 nginx
71beefd3446a4db2cf316c5ca6611256fd77a3e49494e89838c59e520ebfac4c

#查看容器
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS                  NAMES
71beefd3446a        nginx               "nginx -g 'daemon of…"   17 seconds ago      Up 15 seconds               0.0.0.0:8080-
>80/tcp   nginx02

#访问这个容器
59.47.71.220:8080
返回结果：
Welcome to nginx!
```

3.6、查看容器的日志（备注：日志会输出到控制台）

```
#查看容器的日志（备注：日志会输出到控制台）
[root@ansible-server ~]# docker logs nginx02
98.142.138.176 - - [23/May/2019:02:59:08 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36" "-"
2019/05/23 02:59:09 [error] 6#6: *2 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client:
98.142.138.176, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "59.47.71.229:8080", referrer:
"http://59.47.71.229:8080/"
98.142.138.176 - - [23/May/2019:02:59:09 +0000] "GET /favicon.ico HTTP/1.1" 404 556 "http://59.47.71.229:8080/" "Mozilla/5.0
(Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36" "-"

#容器日志保存地址
[root@ansible-server ~]# ls /var/lib/docker/containers/
71beefd3446a4db2cf316c5ca6611256fd77a3e49494e89838c59e520ebfac4c

#进入日志目录，查看访问日志
[root@ansible-server containers]# cd 71beefd3446a4db2cf316c5ca6611256fd77a3e49494e89838c59e520ebfac4c/
[root@ansible-server 71beefd3446a4db2cf316c5ca6611256fd77a3e49494e89838c59e520ebfac4c]# ll
total 28
-rw-r-----. 1 root root 1751 May 23 11:08 71beefd3446a4db2cf316c5ca6611256fd77a3e49494e89838c59e520ebfac4c-json.log
drwx------. 2 root root    6 May 23 10:57 checkpoints
-rw-------. 1 root root 2900 May 23 10:57 config.v2.json
-rw-r--r--. 1 root root 1463 May 23 10:57 hostconfig.json
-rw-r--r--. 1 root root   13 May 23 10:57 hostname
-rw-r--r--. 1 root root  174 May 23 10:57 hosts
drwx------. 3 root root   17 May 23 10:57 mounts
-rw-r--r--. 1 root root   76 May 23 10:57 resolv.conf
-rw-r--r--. 1 root root   71 May 23 10:57 resolv.conf.hash
<br>#查看访问日志
[root@ansible-server 71beefd3446a4db2cf316c5ca6611256fd77a3e49494e89838c59e520ebfac4c]# tail
71beefd3446a4db2cf316c5ca6611256fd77a3e49494e89838c59e520ebfac4c-json.log
{"log":"98.142.138.176 - - [23/May/2019:02:59:08 +0000] \"GET / HTTP/1.1\" 200 612 \"-\" \"Mozilla/5.0 (Windows NT 6.1; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36\" \"-\"\r\n","stream":"stdout","time":"2019-05-
23T02:59:08.9258795Z"}
{"log":"2019/05/23 02:59:09 [error] 6#6: *2 open() \"/usr/share/nginx/html/favicon.ico\" failed (2: No such file or
directory), client: 98.142.138.176, server: localhost, request: \"GET /favicon.ico HTTP/1.1\", host: \"59.47.71.229:8080\",
referrer: \"http://59.47.71.229:8080/\"\r\n","stream":"stdout","time":"2019-05-23T02:59:09.7594971Z"}
{"log":"98.142.138.176 - - [23/May/2019:02:59:09 +0000] \"GET /favicon.ico HTTP/1.1\" 404 556 \"http://59.47.71.229:8080/\"
\"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36\" \"-
```

\"\r\n","stream":"stdout","time":"2019-05-23T02:59:09.7595791Z"}
{"log":"98.142.138.176 - - [23/May/2019:03:08:10 +0000] \"GET / HTTP/1.1\" 304 0 \"-\" \"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36\" \"-\"\r\n","stream":"stdout","time":"2019-05-23T03:08:10.0499976Z"}
{"log":"98.142.138.176 - - [23/May/2019:03:08:12 +0000] \"GET / HTTP/1.1\" 304 0 \"-\" \"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36\" \"-\"\r\n","stream":"stdout","time":"2019-05-23T03:08:12.8716702Z"}
{"log":"98.142.138.176 - - [23/May/2019:03:08:15 +0000] \"GET / HTTP/1.1\" 304 0 \"-\" \"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36\" \"-\"\r\n","stream":"stdout","time":"2019-05-23T03:08:15.1182369Z"}

3.7、--restart=always：指的是服务退出，始终会重启容器

```
#清理所有容器
[root@ansible-server ~]# docker stop $(docker ps -a -q);docker rm $(docker ps -a -q)
4aff4bb376dd
eb53d76f4778
27080338dabb
d5ee27264bd3
b8ecef224d29
4aff4bb376dd
eb53d76f4778
27080338dabb
d5ee27264bd3
b8ecef224d29
<br>#删除所有镜像
```

[root@ansible-server ~]# docker rmi $(docker images -q)

```
#运行容器（--restart=always：指的是服务退出，始终会重启容器）
[root@ansible-server ~]# docker container run -itd -p 8080:80 --name nginx02 --restart=always nginx
1372e859b8e8bff473f2d242a50e0f51f96dabd325800ae3504aabf3e041af55

#查看容器
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS                  NAMES
1372e859b8e8        nginx               "nginx -g 'daemon of…"   5 seconds ago       Up 3 seconds        0.0.0.0:8080-
>80/tcp   nginx02
```

3.8、限制容器使用CPU资源

```
[root@ansible-server ~]# docker container run -itd --cpus 1 --name nginx01 nginx
2d801ef7a76d913124b77e42d14da6722d138501600d076beec1a734642dbf99

[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
2d801ef7a76d        nginx               "nginx -g 'daemon of…"   6 seconds ago       Up 4 seconds        80/tcp
```

3.9、限制内存使用率

```
[root@ansible-server ~]# docker container run -itd --memory 512m --name nginx02 nginx
ea65f58c0e55a38019480c4c75a76e71ee129d310e279e5adea73ac792f1a04e
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
ea65f58c0e55        nginx               "nginx -g 'daemon of…"   8 seconds ago       Up 5 seconds        80/tcp
nginx02
2d801ef7a76d        nginx               "nginx -g 'daemon of…"   4 minutes ago       Up 4 minutes        80/tcp              n
```

4.0、查看容器资源利用率

```
[root@ansible-server ~]# docker container stats nginx02

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT   MEM %               NET I/O             BLOCK
I/O           PIDS
ea65f58c0e55        nginx02             0.00%               1.359MiB / 512MiB   0.27%               648B / 0B           0B /
0B            2

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT   MEM %               NET I/O             BLOCK
I/O           PIDS
ea65f58c0e55        nginx02             0.00%               1.359MiB / 512MiB   0.27%               648B / 0B           0B /
0B            2
```

```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT   MEM %               NET I/O             BLOCK
I/O           PIDS
ea65f58c0e55        nginx02             0.00%               1.359MiB / 512MiB   0.27%               648B / 0B           0B /
0B             2

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT   MEM %               NET I/O             BLOCK
I/O           PIDS
ea65f58c0e55        nginx02             0.00%               1.359MiB / 512MiB   0.27%               648B / 0B           0B /
0B             2

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT   MEM %               NET I/O             BLOCK
I/O           PIDS
ea65f58c0e55        nginx02             0.00%               1.359MiB / 512MiB   0.27%               648B / 0B           0B /
0B             2

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT   MEM %               NET I/O             BLOCK
I/O           PIDS
ea65f58c0e55        nginx02             0.00%               1.359MiB / 512MiB   0.27%               648B / 0B           0B /
0B             2
```

4.1、 查看所有容器IP地址

```
[root@ansible-server ~]# docker inspect --format='{{.Name}} - {{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
$(docker ps -aq)
/composelnmp_nginx_1 -
/thirsty_ptolemy - 172.17.0.3
/composelnmp_php_1 - 172.22.0.2
/composelnmp_mysql_1 - 172.22.0.3
/grafana - 172.23.0.4
/cadvisor - 172.23.0.3
/influxdb - 172.23.0.2
```

四、管理容器常用命令

指令	描述
ls	列出容器
inspect	显示一个或多个容器详细信息
attach	附加本地标准输入，输出和错误到一个运行的容器
exec	在运行容器中执行命令
commit	创建一个新镜像来自一个容器
cp	拷贝文件/文件夹到一个容器
logs	获取一个容器日志
port	列出或指定容器端口映射
stats	显示容器资源使用统计
top	显示一个容器运行的进程
update	更新一个或多个容器配置
stop/start	停止/启动一个或多个容器
rm	删除一个或多个容器

4.1、删除容器和镜像

```
#删除所有容器
docker stop $(docker ps -a -q);docker rm $(docker ps -a -q)

#删除所有镜像
[root@ansible-server ~]# docker rmi -f `docker images -q`
```

4.2、进入容器

```
法一：
[root@ansible-server ~]# docker exec -it nginx02 bash
root@ea65f58c0e55:/# exit

法二：
[root@ansible-server ~]# docker exec -it nginx02 sh
# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var

法三：
```

```
[root@ansible-server ~]# docker run -it centos /bin/bash
```

4.3、commit： 把容器打包成镜像

```
#下载镜像
[root@ansible-server ~]# docker pull centos
Using default tag: latest
latest: Pulling from library/centos
8ba884070f61: Pull complete
Digest: sha256:b40cee82d6f98a785b6ae35748c958804621dc0f2194759a2b8911744457337d
Status: Downloaded newer image for centos:latest
[root@ansible-server ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              9f38484d220f        2 months ago        202MB

#进入镜像
[root@ansible-server ~]# docker run -it centos /bin/bash

#在容器中安装vim
[root@2ffc3f732d99 /]# yum install vim -y
[root@2ffc3f732d99 /]# exit

#查看容器
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
c1e98c693e84        centos              "/bin/bash"         49 seconds ago      Up 46 seconds
dreamy_dirac

#把容器打包成镜像：
[root@ansible-server ~]# docker commit c1e98c693e84 centos-vim
sha256:8377fdf51ec1d46dbb79e96a73e47d72c4be2c4f51ba23969ecea23d0a22b3a8

#查看镜像
[root@ansible-server ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos-vim          latest              8377fdf51ec1        5 seconds ago       202MB
centos              latest              9f38484d220f        2 months ago        202MB
```

4.4、把文件从电脑中拷贝到容器中

```
#查看容器
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
c1e98c693e84        centos              "/bin/bash"         9 minutes ago       Up 9 minutes
dreamy_dirac

#查看需要拷贝的文件
[root@ansible-server ~]# ll
total 294828
-rw-r--r--. 1 root root      2656 May 18 16:26 nginx.conf

#拷贝文件到容器中
[root@ansible-server ~]# docker container cp nginx.conf dreamy_dirac:/root

#进入容器查看，是否有拷贝过来的文件
[root@ansible-server ~]# docker container exec dreamy_dirac ls /root
anaconda-ks.cfg
nginx.conf<br><br>#在容器中的文件重启也不会丢失
```

[root@ansible-server ~]# docker restart dreamy_dirac
dreamy_dirac

4.5、查看容器日志

```
#下载并运行容器
[root@ansible-server ~]# docker container run -itd -p 8080:80 --name nginx02 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
743f2d6c1f65: Pull complete
6bfc4ec4420a: Pull complete
688a776db95f: Pull complete
Digest: sha256:0e409e180983aea7972c92b0a8ae538d1d3c802fe3a8f795ad8049951894590a
Status: Downloaded newer image for nginx:latest
7d5b76c0f8b20038eac1449daba79ded2da7ec41f448ca02252fda877f2ffa9a

#查看容器
```

```
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS                   NAMES
7d5b76c0f8b2        nginx               "nginx -g 'daemon of…"   29 seconds ago      Up 26 seconds       0.0.0.0:8080-
>80/tcp   nginx02
c1e98c693e84        centos              "/bin/bash"              20 minutes ago      Up 4
minutes                                dreamy_dirac

#访问容器的nginx
[root@ansible-server ~]# curl 59.47.71.229:8080
<!DOCTYPE html>
<html>
<body>
<h1>Welcome to nginx!</h1>
</body>
</html>

#查看容器日志
[root@ansible-server ~]# docker logs nginx02
59.47.71.229 - - [23/May/2019:08:32:34 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"
59.47.71.229 - - [23/May/2019:08:32:35 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"
```

4.6、查看容器负载情况

```
[root@ansible-server ~]# docker stats nginx02
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O
BLOCK I/O           PIDS
7d5b76c0f8b2        nginx02             0.00%               1.371MiB / 3.693GiB   0.04%               1.83kB / 2.46kB     0B /
0B              2

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O
BLOCK I/O           PIDS
7d5b76c0f8b2        nginx02             0.00%               1.371MiB / 3.693GiB   0.04%               1.83kB / 2.46kB     0B /
0B              2

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O
BLOCK I/O           PIDS
7d5b76c0f8b2        nginx02             0.00%               1.371MiB / 3.693GiB   0.04%               1.83kB / 2.46kB     0B /
0B              2

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O
BLOCK I/O           PIDS
7d5b76c0f8b2        nginx02             0.00%               1.371MiB / 3.693GiB   0.04%               1.83kB / 2.46kB     0B /
0B              2

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O
BLOCK I/O           PIDS
7d5b76c0f8b2        nginx02             0.00%               1.371MiB / 3.693GiB   0.04%               1.83kB / 2.46kB     0B /
0B              2
```

4.7、查看容器的端口

```
[root@ansible-server ~]# docker port nginx02
80/tcp -> 0.0.0.0:8080
```

4.8、update

```
[root@ansible-server ~]# docker update --help

Usage:  docker update [OPTIONS] CONTAINER [CONTAINER...]

Update configuration of one or more containers

Options:
      --blkio-weight uint16        Block IO (relative weight), between 10 and 1000, or 0
                                   to disable (default 0)
      --cpu-period int             Limit CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int              Limit CPU CFS (Completely Fair Scheduler) quota
      --cpu-rt-period int          Limit the CPU real-time period in microseconds
      --cpu-rt-runtime int         Limit the CPU real-time runtime in microseconds
  -c, --cpu-shares int             CPU shares (relative weight)
      --cpus decimal               Number of CPUs
      --cpuset-cpus string         CPUs in which to allow execution (0-3, 0,1)
      --cpuset-mems string         MEMs in which to allow execution (0-3, 0,1)
      --kernel-memory bytes        Kernel memory limit
  -m, --memory bytes               Memory limit
```

```
      --memory-reservation bytes   Memory soft limit
      --memory-swap bytes          Swap limit equal to memory plus swap: '-1' to enable
                                   unlimited swap
      --restart string             Restart policy to apply when a container exits
```

示例：

```
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS                  NAMES
7d5b76c0f8b2        nginx               "nginx -g 'daemon of…"   22 minutes ago      Up 22 minutes       0.0.0.0:8080-
>80/tcp   nginx02
c1e98c693e84        centos              "/bin/bash"              42 minutes ago      Up 26
minutes                               dreamy_dirac<br><br>#设置容器重启策略
[root@ansible-server ~]# docker update --restart=always 7d5b76c0f8b2
7d5b76c0f8b2
```

五、Volume(数据卷)

5.1、将Docker主机数据挂载到容器

1、保证容器中的数据不丢失

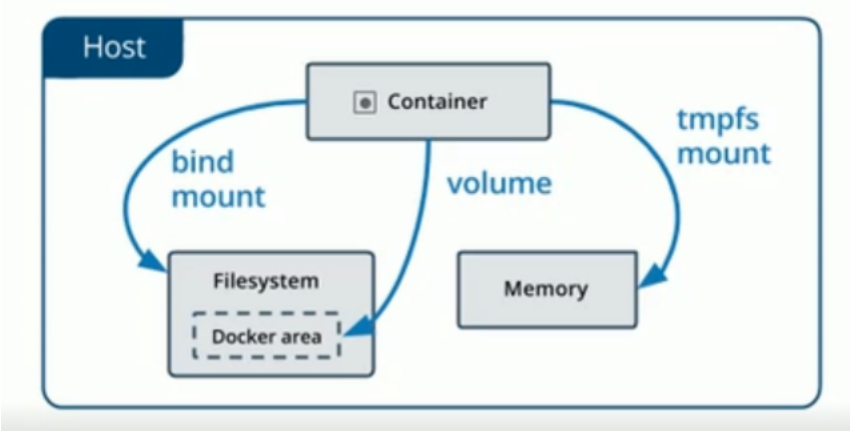
2、原来容器中的数据，不能容器挂了，数据就拿不出来了。

Docker提供三种不同的方式将数据从宿主机挂载到容器中：volumes，bind mounts和tmpfs。

volumes：Docker管理宿主机文件系统的一部分（/var/lib/docker/volumes）。

bind mounts：可以存储在宿主机系统的任意位置。

tmpfs：挂载存储在宿主机系统的内存中，而不会写入宿主机的文件系统。



挂载数据的三种方式：

1、volumes （常用）

[root@ansible-server ~]# ls /var/lib/docker/volumes/
metadata.db

#查看帮助

```
[root@ansible-server ~]# docker volume --help

Usage:  docker volume COMMAND

Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused local volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
```

2、bind mounts (常用)

挂载在宿主机上面

3、tmpfs （不常用）

tmpfs是一种基于内存的文件系统，也叫临时文件系统，tmpfs可以使用RAM,也可以使用swap分区存储。它并不是一个块设备，只要安装就可以使用。是基本RAM的文件系统。

管理卷：

```
# docker volume create nginx-vol
# docker volume ls
# docker volume inspect nginx-vol
```

用卷创建一个容器：

```
# docker run -d -it --name=nginx-test --mount src=nginx-vol,dst=/usr/share/nginx/html nginx
# docker run -d -it --name=nginx-test -v nginx-vol:/usr/share/nginx/html nginx
```

清理：

```
# docker container stop nginx-test
# docker container rm nginx-test
# docker volume rm nginx-vol
```

注意：

1. 如果没有指定卷，自动创建。
2. 建议使用—mount，更通用。

官方文档：https://docs.docker.com/engine/admin/volumes/volumes/#start-a-container-with-a-volume

5.2、创建挂载卷，并挂载到容器中

```
#创建卷
[root@ansible-server ~]# docker volume create nginx-vo1
nginx-vo1

#卷存储位置
[root@ansible-server ~]# docker volume ls
DRIVER              VOLUME NAME
local               nginx-vo1

#查看卷的详细信息
[root@ansible-server ~]# docker volume inspect nginx-vo1
[
    {
        "CreatedAt": "2019-05-23T17:09:33+08:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/nginx-vo1/_data",
        "Name": "nginx-vo1",
        "Options": {},
        "Scope": "local"
    }
]

#挂载卷
语法：
docker run -itd, nginx的名称,--mount src=挂载的数据卷, dst=nginx网站的根目录
[root@ansible-server ~]# docker run -itd --name=nginx-test --mount src=nginx-vo1,dst=/usr/share/nginx/html nginx
ef9087684fb9ffe8775895215e0bbb13d1a8ad39f5b30c93a171213522df10c5

#查看容器
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
ef9087684fb9        nginx               "nginx -g 'daemon of…"   3 minutes ago       Up 3 minutes
80/tcp              nginx-test
7d5b76c0f8b2        nginx               "nginx -g 'daemon of…"   45 minutes ago      Exited (0) 22 minutes
ago                       nginx02
c1e98c693e84        centos              "/bin/bash"              About an hour ago   Up About an
hour                                dreamy_dirac

#进入容器
[root@ansible-server ~]# docker exec -it nginx-test bash
root@ef9087684fb9:/#

#进入nginx网站根目录
root@ef9087684fb9:/# cd /usr/share/nginx/html/

#创建一个文件
root@ef9087684fb9:/usr/share/nginx/html# touch 1.html
```

```
root@ef9087684fb9:/usr/share/nginx/html# ls
1.html  50x.html  index.html

#查看容器外面宿主机目录中是否有这几个文件（备注：再开一个终端）
[root@ansible-server ~]# cd /var/lib/docker/volumes/nginx-vo1/_data/
[root@ansible-server _data]# ls
1.html  50x.html  index.html

[root@ansible-server _data]# exit
```

5.3、演示容器删除了，数据卷的数据还在

(备注：如果在容器中删除了数据，那么本地也会跟着删除)

```
#创建10个文件
root@ef9087684fb9:/usr/share/nginx/html# touch {1..10}.txt

root@ef9087684fb9:/usr/share/nginx/html# ls
1.txt  10.txt  2.txt  3.txt  4.txt  5.txt  50x.html  6.txt  7.txt  8.txt  9.txt  index.html

root@ef9087684fb9:/usr/share/nginx/html# exit
exit

#删除所有容器
[root@ansible-server ~]# docker rm -f $(docker ps -q -a)
ef9087684fb9
7d5b76c0f8b2
c1e98c693e84

#查看容器
[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES

#查看数据卷
[root@ansible-server _data]# cd /var/lib/docker/volumes/nginx-vo1/_data
[root@ansible-server _data]# ls
10.txt  1.txt  2.txt  3.txt  4.txt  50x.html  5.txt  6.txt  7.txt  8.txt  9.txt  index.html
```

5.4、运行容器，增加端口，再通过本地数据卷中写入一个a.html的文件。再用浏览器访问他看是否可以打开

```
#再运行容器，增加端口
[root@ansible-server ~]# docker run -itd --name=nginx-test -p 8080:80 --mount src=nginx-vo1,dst=/usr/share/nginx/html nginx
7c82062a7435997717a938a252684ee5832caafe089d97ebf3948010951ae18a
[root@ansible-server ~]# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
7c82062a7435 nginx "nginx -g 'daemon of…" 4 seconds ago Up 2 seconds 0.0.0.0:8080->80/tcp nginx-test

#容器数据卷存储目录
[root@ansible-server _data]# cd /var/lib/docker/volumes/nginx-vo1/_data
[root@ansible-server _data]# ls
10.txt 1.txt 2.txt 3.txt 4.txt 50x.html 5.txt 6.txt 7.txt 8.txt 9.txt index.html

#创建一个html的文件
[root@ansible-server _data]# vi a.html
<h>welcome nulige</n>

#通过浏览器进行访问
http://59.47.71.229:8080/a.html
返回结果：
welcome nulige
```

5.5、实现数据卷共享他们的数据

```
#数据卷可以共享他们的数据
[root@ansible-server ~]# docker run -itd --name=nginx-test -p 8080:80 --mount src=nginx-vo1,dst=/usr/share/nginx/html nginx
7c82062a7435997717a938a252684ee5832caafe089d97ebf3948010951ae18a

#本地数据卷目录
[root@ansible-server _data]# cd /var/lib/docker/volumes/nginx-vo1/_data

[root@ansible-server _data]# ls
50x.html index.html

#创建一个文件
[root@ansible-server _data]# vi a.html
<h>welcome nulige</n>
```

```
#查看文件
[root@ansible-server _data]# ls
50x.html a.html  index.html

#通过再创建一个容器，共享上面这个数据卷。实现了数据共享，多个nginx服务都访问同一个数据卷。这里有点像nginx访问nfs共享文件夹的功能。
[root@ansible-server ~]# docker run -itd --name=nginx-test02 -p 8081:80 --mount src=nginx-vo1,dst=/usr/share/nginx/html nginx
6e984d5e99a43487e616e92d75a9e9c57c4da3a0d7f0833c37f46072c3673970

[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS                   NAMES
6e984d5e99a4        nginx               "nginx -g 'daemon of…"   16 seconds ago      Up 14 seconds       0.0.0.0:8081-
>80/tcp   nginx-test02
7c82062a7435        nginx               "nginx -g 'daemon of…"   43 minutes ago      Up 43 minutes       0.0.0.0:8080-
>80/tcp   nginx-test
```

5.6、没有指定数据卷，他默认会创建数据库

```
#没有指定数据卷，他默认会自动创建数据卷
卷分类：
一种是：命名卷，自己指定的卷
一种是：匿名卷，系统默认创建的卷
<br>#创建匿名卷

[root@ansible-server ~]# docker run -itd --name=nginx-test03 -p 8082:80 --mount src=,dst=/usr/share/nginx/html nginx
bcbdee85ae6d005971649a802484dacc8b6aca6f96adf70ef4c4ba3dab32bee9 <br>#查看卷
[root@ansible-server ~]# docker volume ls
DRIVER              VOLUME NAME
local               92dbb978ec9d767fee49b1cdb440cd45ada36feb42b064132ba152150780fe67 #匿名卷
local               nginx-vo1  #命名卷
```

5.7、通过bind挂载数据卷

示例：

```
#删除容器
[root@ansible-server ~]# docker rm -f $(docker ps -q -a)
bcbdee85ae6d
6e984d5e99a4
7c82062a7435

#绑定系统中已经存在的卷,src=app，这个目录必须存在，否则会报错
加参数：type=bind

先创建一个文件夹
[root@ansible-server ~]# mkdir -p /app/wwwroot

再绑定到这个文件夹中
[root@ansible-server ~]# docker run -itd --name=nginx-test --mount type=bind,src=/app/wwwroot,dst=/usr/share/nginx/html nginx
02c1bdaab564b3add7bbe0ac9b289e616e4bc40c965ff9824188bb99ed1fccb2

[root@ansible-server ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
02c1bdaab564        nginx               "nginx -g 'daemon of…"   5 seconds ago       Up 3 seconds        80/tcp
nginx-test

#进入容器
[root@ansible-server ~]# docker exec -it nginx-test bash
```

```
#查看mount是否挂载到/usr/share/nginx/html目录
root@02c1bdaab564:/# mount
overlay on / type overlay
(rw,relatime,seclabel,lowerdir=/var/lib/docker/overlay2/l/3QG6SC3NFVC6UJJQNVQGSHSCJV:/var/lib/docker/overlay2/l/IDJTHZBOQVLBQ7
TISUNIWAAKS4:/var/lib/docker/overlay2/l/57TQY5XX7CMEROHBTYSQYNQTFY:/var/lib/docker/overlay2/l/FBBYWNRKKGDXOBECYPD3KFPUZN,upper
dir=/var/lib/docker/overlay2/e7815303046019f7b9a48c1b1549bd46d25a5011aded5c61a4ca11620880f91d/diff,workdir=/var/lib/docker/ove
rlay2/e7815303046019f7b9a48c1b1549bd46d25a5011aded5c61a4ca11620880f91d/work)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev type tmpfs (rw,nosuid,seclabel,size=65536k,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,seclabel,gid=5,mode=620,ptmxmode=666)
sysfs on /sys type sysfs (ro,nosuid,nodev,noexec,relatime,seclabel)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,relatime,seclabel,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup
(ro,nosuid,nodev,noexec,relatime,seclabel,xattr,release_agent=/usr/lib/systemd/systemd-cgroups-agent,name=systemd)
cgroup on /sys/fs/cgroup/perf_event type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,perf_event)
cgroup on /sys/fs/cgroup/blkio type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,blkio)
cgroup on /sys/fs/cgroup/cpuset type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,cpuset)
cgroup on /sys/fs/cgroup/memory type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,memory)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,cpuacct,cpu)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,net_prio,net_cls)
cgroup on /sys/fs/cgroup/devices type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,devices)
cgroup on /sys/fs/cgroup/pids type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,pids)
cgroup on /sys/fs/cgroup/freezer type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,freezer)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (ro,nosuid,nodev,noexec,relatime,seclabel,hugetlb)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime,seclabel)
/dev/mapper/centos-root on /etc/resolv.conf type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/centos-root on /etc/hostname type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/mapper/centos-root on /etc/hosts type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
shm on /dev/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,seclabel,size=65536k)
/dev/mapper/centos-root on /usr/share/nginx/html type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
devpts on /dev/console type devpts (rw,nosuid,noexec,relatime,seclabel,gid=5,mode=620,ptmxmode=666)
proc on /proc/bus type proc (ro,relatime)
proc on /proc/fs type proc (ro,relatime)
proc on /proc/irq type proc (ro,relatime)
proc on /proc/sys type proc (ro,relatime)
proc on /proc/sysrq-trigger type proc (ro,relatime)
tmpfs on /proc/acpi type tmpfs (ro,relatime,seclabel)
tmpfs on /proc/kcore type tmpfs (rw,nosuid,seclabel,size=65536k,mode=755)
tmpfs on /proc/keys type tmpfs (rw,nosuid,seclabel,size=65536k,mode=755)
tmpfs on /proc/timer_list type tmpfs (rw,nosuid,seclabel,size=65536k,mode=755)
tmpfs on /proc/timer_stats type tmpfs (rw,nosuid,seclabel,size=65536k,mode=755)
tmpfs on /proc/sched_debug type tmpfs (rw,nosuid,seclabel,size=65536k,mode=755)
tmpfs on /proc/scsi type tmpfs (ro,relatime,seclabel)
tmpfs on /sys/firmware type tmpfs (ro,relatime,seclabel)

#备注：通过bind挂载的容器，他会隐藏掉文件夹下面的内容
root@02c1bdaab564:/# cd /usr/share/nginx/html/
root@02c1bdaab564:/usr/share/nginx/html# ls

#再开一个终端
[root@ansible-server _data]# cd /app/wwwroot/

#往里面写入点内容
[root@ansible-server wwwroot]# echo "<h>welcome nulige </h>" >index.html
[root@ansible-server wwwroot]# cat index.html
<h>welcome nulige</h>

#再到容器中查看
[root@ansible-server ~]# docker exec -it nginx-test bash

root@02c1bdaab564:/# cd /usr/share/nginx/html/

root@02c1bdaab564:/usr/share/nginx/html# ls
index.html

root@02c1bdaab564:/usr/share/nginx/html# cat index.html
<h>welcome nulige </h>
```

#bind的应用场景

1、用在tomcat的部署中，直接生成jar包，挂载到部署目录中。

```
ls target/xxx.war jar
```

六、应用案例

6.1、搭建LNMP网站平台实战

一、搭建LNMP网站平台（nginx+php+mysql）

1、创建网络

```
[root@ansible-server ~]# docker network create lnmp
#查看网络
[root@ansible-server ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
b892a0991d9d        bridge              bridge              local
f5f547cc8686        host                host                local
35b5589b014b        lnmp                bridge              local
a480fd78ff5a        none                null                local
```

2、创建Mysql数据库容器

```
docker run -itd \
--name lnmp_mysql \
--net lnmp \
-p 3306:3306 \
--mount src=mysql-vol,dst=/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=123456 \
mysql --character-set-server=utf8

#查看数据卷
[root@ansible-server ~]# docker volume ls
DRIVER              VOLUME NAME
local               92dbb978ec9d767fee49b1cdb440cd45ada36feb42b064132ba152150780fe67
local               mysql-vol
local               nginx-vo1

#数据卷目录存放地址
[root@ansible-server ~]# ls /var/lib/docker/volumes/mysql-vol/_data/
auto.cnf       ca-key.pem       ib_buffer_pool  ibtmp1        performance_schema  server-key.pem
binlog.000001  ca.pem           ibdata1         #innodb_temp  private_key.pem     sys
binlog.000002  client-cert.pem  ib_logfile0     mysql         public_key.pem      undo_001
binlog.index   client-key.pem   ib_logfile1     mysql.ibd     server-cert.pem     undo_002

#查看lnmp_mysql日志
[root@ansible-server ~]# docker logs lnmp_mysql

#查看top
[root@ansible-server ~]# docker top lnmp_mysql
UID                 PID                 PPID                C                   STIME               TTY
TIME                CMD
polkitd             5896                5879                2                   11:07               pts/0
00:00:08            mysqld --character-set-server=utf8
```

3、创建所需数据库

```
[root@ansible-server ~]# docker exec lnmp_mysql sh -c 'exec mysql -uroot -p"$MYSQL_ROOT_PASSWORD" -e "create database
wordpress"'
#提示密码不安全
mysql: [Warning] Using a password on the command line interface can be insecure.

#查看数据库
[root@ansible-server ~]# docker exec lnmp_mysql sh -c 'exec mysql -uroot -p"123456" -h 59.47.71.229 -e "show databases;"'
mysql: [Warning] Using a password on the command line interface can be insecure.
Database
information_schema
mysql
performance_schema
sys
wordpress
```

4、创建PHP环境容器(备注：镜像中包括：nginx+php-fpm)

```
docker run -itd \
--name lnmp_web \
--net lnmp \
-p 88:80 \
--mount type=bind,src=/app/wwwroot,dst=/var/www/html richarvey/nginx-php-fpm

#查看容器
[root@ansible-server wordpress]# docker ps -a
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS
PORTS                                  NAMES
6de0d790b079        richarvey/nginx-php-fpm    "docker-php-entrypoi…"   9 minutes ago       Up 9 minutes        443/tcp,
9000/tcp, 0.0.0.0:88->80/tcp   lnmp_web
```

```
cee253e2ef97        mysql                    "docker-entrypoint.s…"   12 minutes ago      Up 12 minutes       0.0.0.0:3306-
>3306/tcp, 33060/tcp       lnmp_mysql
```

5、以wordpress博客为例测试

```
wget https://cn.wordpress.org/wordpress-4.7.4-zh_CN.tar.gz
tar zxf wordpress-4.7.4-zh_CN.tar.gz -C /app/wwwroot

#查看解压文件
[root@ansible-server ~]# ls /app/wwwroot/
index.html  wordpress

[root@ansible-server wordpress]# iptables -I INPUT -s 0.0.0.0/0 -d 0.0.0.0/0 -j ACCEPT
[root@ansible-server wordpress]# service iptables save
[root@ansible-server wordpress]# service iptables restart
```

6、浏览器测试访问

```
http://59.47.71.229:88/wordpress

#访问后会跳转到
http://59.47.71.229:88/wordpress/wp-admin/setup-config.php
```

#填写信息

参考：

https://ke.qq.com/course/366769?taid=2769884539099313&dialog=1

分类： docker

标签： docker命令详细讲解

努力哥
关注 - 43
粉丝 - 480
+加关注

0 0

posted @ 2019-05-23 10:14 努力哥 阅读(200) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。