参考网址：<http://www.centoscn.com/image-text/install/2014/1128/4202.html>

Docker是一个开源的应用容器引擎，可以轻松的为任何应用创建一个轻量级的、可移植的、自给自足的容器。利用Linux的LXC、AUFS、Go语言、cgroup实现了资源的独立，可以很轻松的实现文件、资源、网络等隔离，其最终的目标是实现类似PaaS平台的应用隔离。

    Docker值得关注的特性：

文件系统隔离：每个进程容器运行在一个完全独立的根文件系统里。

资源隔离：系统资源，像CPU和内存等可以分配到不同的容器中，使用cgroup。

网络隔离：每个进程容器运行在自己的网络空间，虚拟接口和IP地址。

日志记录：Docker将会收集和记录每个进程容器的标准流（stdout/stderr/stdin），用于实时检索或批量检索。

变更管理：容器文件系统的变更可以提交到新的映像中，并可重复使用以创建更多的容器。无需使用模板或手动配置。

交互式shell：Docker可以分配一个虚拟终端并关联到任何容器的标准输入上，例如运行一个一次交互shell。

**Docker通常用于如下场景：**

web应用的自动化打包和发布；

自动化测试和持续集成、发布

在服务型环境中部署和调整数据库或其他的后台应用；

从头编译或者扩展现有的OpenShift或Cloud Foundry平台来搭建自己的PaaS环境。

     本文介绍如何在RedHat/CentOS环境下安装Docker。官方文档要求Linux kernel至少3.8以上，且docker只能运行在64位的系统中。由于RHEL6和CentOS6的内核版本为2.6，因此必须要先升级内核。

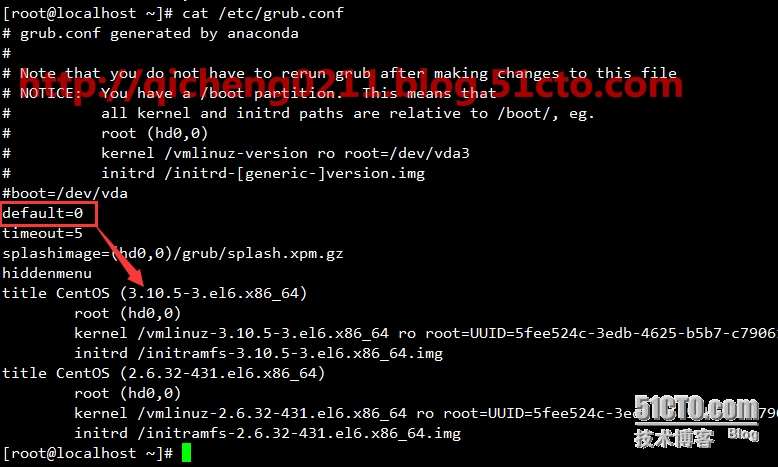
    下面以CentOS6.5（64位）为例，介绍下docker安装步骤和使用方法：

**一、升级内核（带aufs模块）**

    1、yum安装带aufs模块的3.10内核（或到这里下载kernel手动安装：http://down.51cto.com/data/1903250）

|  |
| --- |
| cd /etc/yum.repos.d  wget  http://www.hop5.**in**/yum/el6/hop5.repo  yum  install  kernel-ml-aufs kernel-ml-aufs-devel |

    2、修改grub的主配置文件/etc/grub.conf，设置default=0，表示第一个title下的内容为默认启动的kernel（一般新安装的内核在第一个位置）。



    3、重启系统，这时候你的内核就成功升级了。

|  |
| --- |
| [root@localhost ~]# uname -r  3.10.5-3.el6.x86\_64 |

    查看内核是否支持aufs：

|  |
| --- |
| [root@localhost ~]# grep aufs /proc/filesystems  nodev    aufs |

**二、安装docker**

    1、首先关闭selinux：

|  |
| --- |
| setenforce 0  sed -i '/^SELINUX=/c\SELINUX=disabled' /etc/selinux/config |

    2、在Fedora EPEL源中已经提供了docker-io包，下载安装epel：

|  |
| --- |
| rpm -ivh http://mirrors.sohu.com/fedora-epel/6/x86\_64/epel-release-6-8.noarch.rpm  sed -i 's/^mirrorlist=https/mirrorlist=http/' /etc/yum.repos.d/epel.repo |

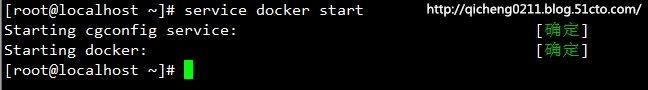
    3、yum安装docker-io：

|  |  |
| --- | --- |
|  | yum -y install docker-io |



    4、启动docker：

|  |
| --- |
| service docker start |
| [root@master ~]# service docker start  Starting cgconfig service: Error: cannot mount cpuset to /cgroup/cpuset: Device or resource busy  /sbin/cgconfigparser; error loading /etc/cgconfig.conf: Cgroup mounting failed  Failed to parse /etc/cgconfig.conf or /etc/cgconfig.d [FAILED]  Starting docker: [ OK ]  出现这个错误的解决办法：  <http://stackoverflow.com/questions/25183063/docker-on-rhel-6-cgroup-mounting-failing>  【办法】  I have the same issue.  IMG_256  (1) check cgconfig status  # /etc/init.d/cgconfig status  if it stopped, restart it  # /etc/init.d/cgconfig restart  check cgconfig is running  IMG_257  (2) check cgconfig is on  # chkconfig --list cgconfig  cgconfig 0:off 1:off 2:off 3:off 4:off 5:off 6:off  if cgconfig is off, turn it on  IMG_258  (3) if still does not work, may be some cgroups modules is missing. In the kernel .config file, make menuconfig, add those modules into kernel and recompile and reboot  after that, it should be OK  IMG_259   |  | | --- | | [share](http://stackoverflow.com/a/30993928)[improve this answer](http://stackoverflow.com/posts/30993928/edit) | |



    5、查看docker版本：



[root@master ~]# docker version

Client version: 1.7.1

Client API version: 1.19

Go version (client): go1.4.2

Git commit (client): 786b29d/1.7.1

OS/Arch (client): linux/amd64

Server version: 1.7.1

Server API version: 1.19

Go version (server): go1.4.2

Git commit (server): 786b29d/1.7.1

OS/Arch (server):

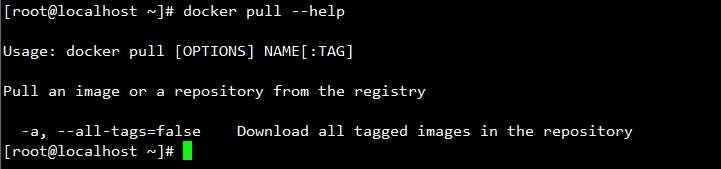
    查看docker日志：

|  |  |
| --- | --- |
|  | cat /var/log/docker  [root@master Desktop]# cat /var/log/docker  \nFri Nov 4 09:30:29 PDT 2016\n  time="2016-11-04T09:30:30.615758467-07:00" level=warning msg="You are running linux kernel version 2.6.32-642.el6.x86\_64, which might be unstable running docker. Please upgrade your kernel to 3.10.0." |

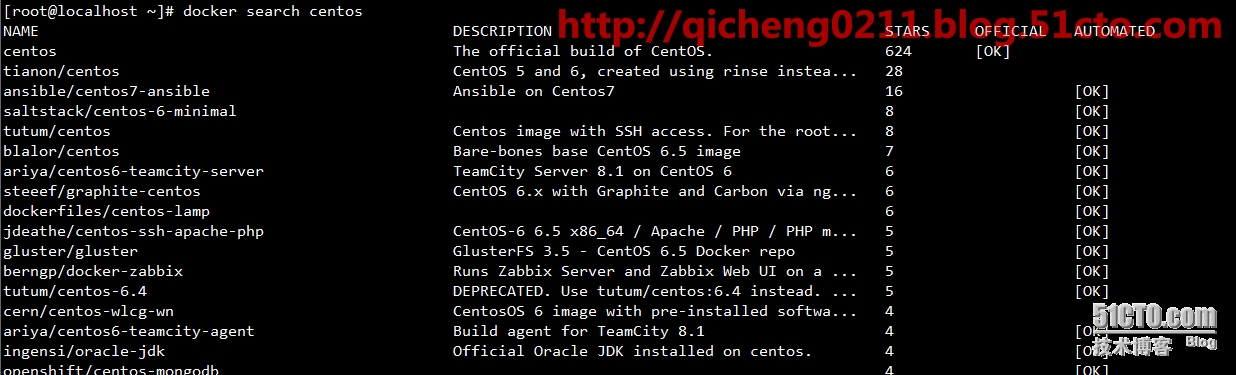
**三、docker命令的使用**

    1、直接输入docker命令来查看所有的**Options和Commands**。

    查看某一个command的详细使用方法：docker COMMAND --help



    2、搜索可用的docker镜像：docker search NAME



    3、下载镜像：docker pull NAME[:TAG]

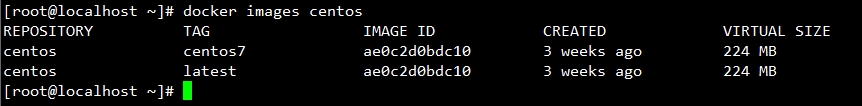
    比如获取最新的centos镜像：docker pull centos:latest

注意：这里要写用docker search搜索到的完整的镜像名。  
    4、查看安装的镜像：docker images [NAME]

[root@master ~]# docker images busybox

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

busybox latest 9967c5ad88de 6 weeks ago 1.093 MB



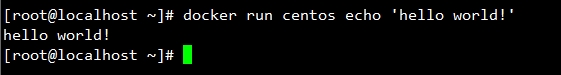
    5、在docker容器中运行命令：docker run IMAGE [COMMAND] [ARG...]

    docker run命令有两个参数，一个是镜像名，一个是要在镜像中运行的命令。

注意：IMAGE=REPOSITORY[:TAG]，如果IMAGE参数不指定镜像的TAG，默认TAG为latest。

[root@master ~]# docker run busybox echo Hello World

Hello World

    在刚刚下载的镜像中输出"hello word"：docker run centos echo 'hello world!'  


    6、列出容器：docker ps -a

查看最近生成的容器：docker ps -l

[root@master ~]# docker ps -l

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f75f388773d2 busybox "echo Hello World" About a minute ago Exited (0) About a minute ago cocky\_brattain

由返回结果可见还包括了最近一次运行的命令

查看正在运行的容器：docker ps

[root@master ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

    7、显示容器的标准输出：docker logs CONTAINERID

无需拷贝完整的id，一般写最开始的三至四个字符即可。

[root@master ~]# docker ps -l

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f75f388773d2 busybox "echo Hello World" About a minute ago Exited (0) About a minute ago cocky\_brattain

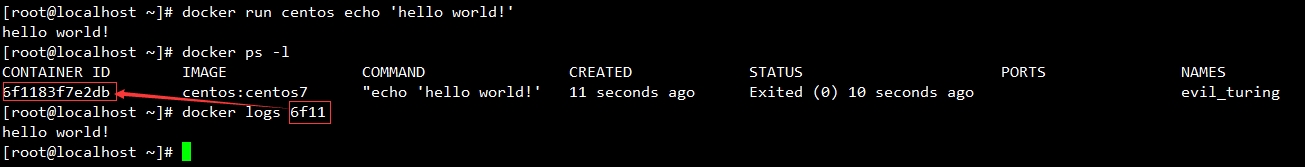
[root@master ~]# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

busybox latest 9967c5ad88de 6 weeks ago 1.093 MB

[root@master ~]# docker logs f75

Hello World #这是最近一次命令执行的返回结果



    8、在容器中安装新程序，比如安装ifconfig命令（centos7默认没有ifconfig）：docker run centos yum install net-tools -y #与平常的命令无异

    如果yum不指定-y参数的话，yum命令会进入交互模式，需要用户输入命令来进行确认，在docker环境中是无法响应这种交互的。但使用docker run的-i -t参数就会响应这种交互，用户可以输入命令了，比如：docker run -i -t centos yum install net-tools

    9、保存对容器的修改并生成新的镜像：docker commit CONTAINERID [REPOSITORY[:TAG]]

REPOSITORY参数可以是新的镜像名字，也可以是旧的镜像名；如果和旧的镜像名和TAG都相同，会覆盖掉旧的镜像。

[root@master ~]# docker commit f75f388773d2 busybox:new

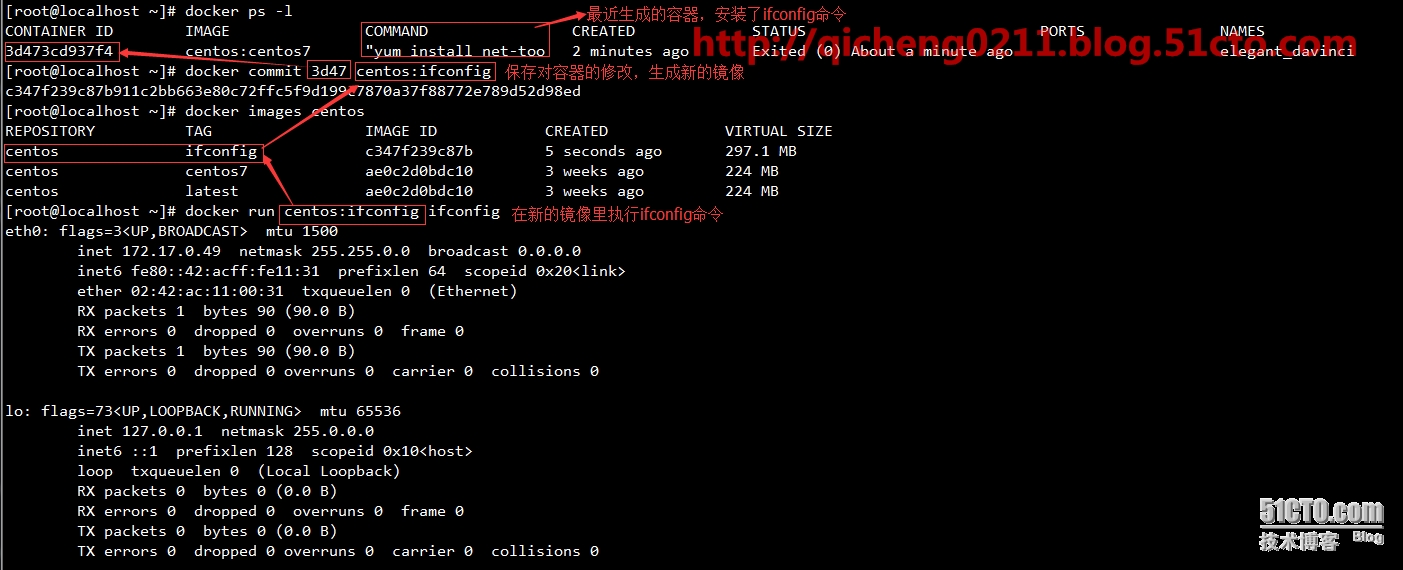
4a435d816e2bea471b1d6ad982037d9738797623a75ab381bf17a466cb387da3

[root@master ~]# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

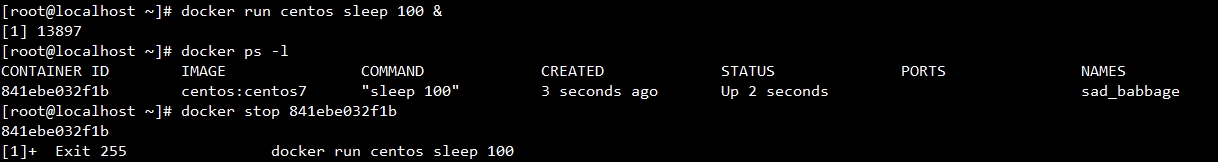
busybox new 4a435d816e2b 5 seconds ago 1.093 MB

busybox latest 9967c5ad88de 6 weeks ago 1.093 MB



    10、停止正在运行的容器：docker stop CONTAINERID

    默认等待10秒钟再杀死指定容器。可以使用-t参数来设置等待时间。



    11、查看容器或镜像的详细信息：docker inspect CONTAINERID|IMAGE

参数可以是容器的ID或者是镜像名（NAME:TAG）。

[root@master ~]# docker inspect busybox:new

[

{

"Id": "4a435d816e2bea471b1d6ad982037d9738797623a75ab381bf17a466cb387da3",

"Parent": "9967c5ad88de8c101809f7f22d4774b6791fe46ac3033d57abf7ebb1dd8e36ee",

"Comment": "",

"Created": "2016-11-19T14:09:10.919471902Z",

"Container": "f75f388773d29d6df28e352a2892380da632020b35cb496b74c9ffc3b8bb0c17",

"ContainerConfig": {

"Hostname": "f75f388773d2",

"Domainname": "",

"User": "",

"AttachStdin": false,

"AttachStdout": true,

"AttachStderr": true,

"PortSpecs": null,

"ExposedPorts": null,

"Tty": false,

"OpenStdin": false,

"StdinOnce": false,

"Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

],

"Cmd": [

"echo",

"Hello",

"World"

],

"Image": "busybox",

"Volumes": null,

"VolumeDriver": "",

"WorkingDir": "",

"Entrypoint": null,

"NetworkDisabled": false,

"MacAddress": "",

"OnBuild": null,

"Labels": {}

},

"DockerVersion": "1.7.1",

"Author": "",

"Config": {

"Hostname": "",

"Domainname": "",

"User": "",

"AttachStdin": false,

"AttachStdout": false,

"AttachStderr": false,

"PortSpecs": null,

"ExposedPorts": null,

"Tty": false,

"OpenStdin": false,

"StdinOnce": false,

"Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

],

"Cmd": [

"echo",

"Hello",

"World"

],

"Image": "",

"Volumes": null,

"VolumeDriver": "",

"WorkingDir": "",

"Entrypoint": null,

"NetworkDisabled": false,

"MacAddress": "",

"OnBuild": null,

"Labels": {}

},

"Architecture": "amd64",

"Os": "linux",

"Size": 0,

"VirtualSize": 1093484

}

]

[root@master ~]#



    12、删除容器：docker rm CONTAINERID

    查看所有容器ID：docker ps -a -q

    删除所有的容器：docker rm $(docker ps -a -q)

13、删除镜像：docker rmi IMAGE

[root@master ~]# docker rmi busybox:new

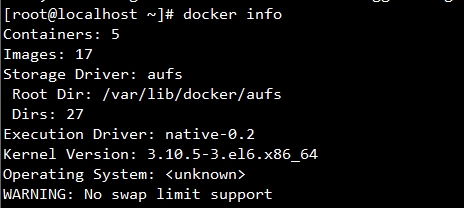
Untagged: busybox:new

Deleted: 4a435d816e2bea471b1d6ad982037d9738797623a75ab381bf17a466cb387da3



14.查看docker的信息，包括Containers和Images数目、kernel版本等。

docker info



**四、创建容器并登入的操作**

    1、创建一个新容器并登入：docker run -i -t IMAGE /bin/bash

    使用image创建container并进入交互模式,login shell是/bin/bash，现在可以自由的对容器进行操作了。最后使用exit退出容器。

注意：如果IMAGE参数不指定TAG，默认TAG为latest。

[root@da09cd751694 tmpfiles]# echo hello Docker >>test.txt

[root@da09cd751694 tmpfiles]# more test.txt

This is the first file...

hello Docker

    2、启动一个退出的容器：docker start CONTAINERID

    3、attach到运行中的容器：docker attach CONTAINERID



[root@master ~]# docker attach da

You cannot attach to a stopped container, start it first

[root@master ~]# docker start da

da

[root@master ~]# docker ps -l

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

da09cd751694 centos "/bin/bash" 16 minutes ago Up 6 seconds cocky\_pasteur

[root@master ~]# docker attach da

^C

[root@da09cd751694 /]# ^C

//==================20171031