

**第一章 Docker简介**

Docker官网: <https://www.docker.com/>

Docker中文社区: <http://www.docker.org.cn/>

Docker HUB: <https://hub.docker.com/>

* 1. **Docker 定义**

Docker 提供了一个可以运行你的应用程序的封套(envelope)，或者说容器它原本是 dotCloud 启动的一个业余项目，并在前些时候开源了。它吸引了大量的关注和讨论，导致 dotCloud 把它重命名到 Docker Inc。它最初是用 Go 语言编写的，它就相当于是加在 LXC（LinuX Containers，linux 容器）上的管道，允许开发者在更高层次的概念上工作。

Docker也是一个云计算平台，它利用Linux的LXC、AUFU、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 会像一个可移植的容器引擎那样工作。它把应用程序及所有程序的依赖环境打包到一个虚拟容器中，这个虚拟容器可以运行在任何一种 Linux 服务器上。这大大地提高了程序运行的灵活性和可移植性，无论需不需要许可、是在公共云还是私密云、是不是裸机环境等等。

* 1. **Docker与虚拟机区别**

|  |  |
| --- | --- |
| **传统虚拟机** | **Docker** |
| 依赖物理CPU和内存，是硬件级别的。 | 在操作系统上，利用操作系统的containerization( 集装箱化) 技术，可以在虚拟机上运行。 |
| 一般都是指操作系统镜像，比较复杂。 | docker开源而且轻量，称为“容器”，单个容器适合部署少量应用。 |
| 使用快照来保存状态。 | 引入了类似源代码管理机制，将容器的快照历史版本一一记录。 |
| 在构建系统的时候较为复杂，需要大量的人力。 | docker可以通过Dockfile来构建整个容器，重启和构建速度很快。更重要的是Dockfile可以手动编写，这样应用程序开发人员可以通过发布Dockfile来指导系统环境和依赖，这样对于持续交付十分有利。 |

* 1. **Docker特性**

|  |  |
| --- | --- |
| **文件系统隔离** | 每个进程容器运行在完全独立的根文件系统里 |
| **资源隔离** | 可以使用cgroup为每个进程容器分配不同的系统资源，例如CPU和内存 |
| **网络隔离** | 每个进程容器运行在自己的网络命名空间里，拥有自己的虚拟接口和IP地址 |
| **写时复制** | 采用写时复制方式创建根文件系统，这让部署变得极其快捷，并且节省内存和硬盘空间。 |
| **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，例如运行一个一次性交互shell。 |

* 1. **Docker 工作原理**

Docker 使用客户端-服务器 (C/S) 架构模式。Docker 客户端会与 Docker 守护进程进行通信。Docker 守护进程会处理复杂繁重的任务，例如建立、运行、发布你的 Docker 容器。Docker 客户端和守护进程可以运行在同一个系统上，当然你也可以使用 Docker 客户端去连接一个远程的 Docker 守护进程。Docker 客户端和守护进程之间通过 socket 或者 RESTful API 进行通信。

|  |  |
| --- | --- |
| **Docker 守护进程** | 如图所示，Docker 守护进程运行在一台主机上。用户并不直接和守护进程进行交互，而是通过 Docker 客户端间接和其通信。 |
| **Docker 客户端** | Docker 客户端，实际上是 docker 的二进制程序，是主要的用户与 Docker 交互方式。它接收用户指令并且与背后的 Docker 守护进程通信，如此来回往复。 |

* 1. **Docker内部组件**

Docker 镜像 - Docker images

Docker 仓库 - Docker registeries

Docker 容器 - Docker containers

|  |  |
| --- | --- |
| **Docker 镜像** | Docker 镜像是 Docker 容器运行时的只读模板，每一个镜像由一系列的层 (layers) 组成。Docker 使用 UnionFS 来将这些层联合到单独的镜像中。UnionFS 允许独立文件系统中的文件和文件夹(称之为分支)被透明覆盖，形成一个单独连贯的文件系统。正因为有了这些层的存在，Docker 是如此的轻量。 |
| 当你改变了一个 Docker 镜像，比如升级到某个程序到新的版本，一个新的层会被创建。因此，不用替换整个原先的镜像或者重新建立(在使用虚拟机的时候你可能会这么做)，只是一个新 的层被添加或升级了。现在你不用重新发布整个镜像，只需要升级，层使得分发 Docker 镜像变得简单和快速。 | |
| **Docker 仓库** | Docker 仓库用来保存镜像，可以理解为代码控制中的代码仓库。同样的，Docker 仓库也有公有和私有的概念。公有的 Docker 仓库名字是 Docker Hub。Docker Hub 提供了庞大的镜像集合供使用。这些镜像可以是自己创建，或者在别人的镜像基础上创建。Docker 仓库是 Docker 的分发部分。 |
| **Docker 容器** | Docker 容器和文件夹很类似，一个Docker容器包含了所有的某个应用运行所需要的环境。每一个 Docker 容器都是从 Docker 镜像创建的。Docker 容器可以运行、开始、停止、移动和删除。每一个 Docker 容器都是独立和安全的应用平台，Docker 容器是 Docker 的运行部分。 |



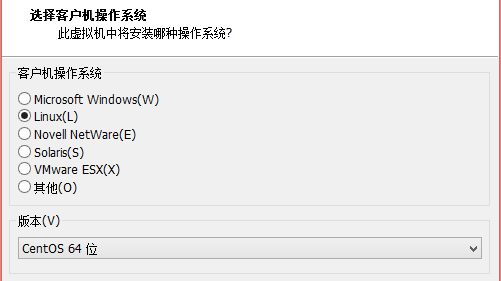
**第二章 Docker安装**

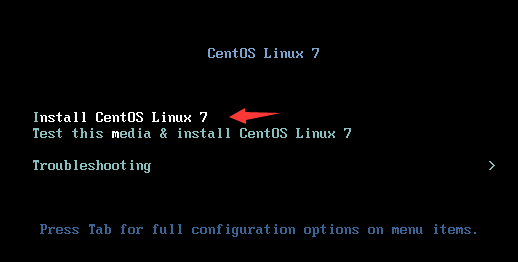
Docker官网: <https://www.docker.com/>

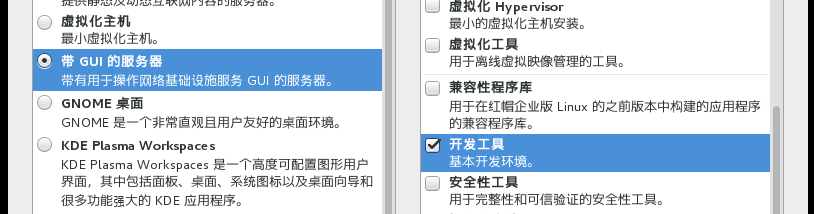
Docker中文社区: <http://www.docker.org.cn/>

Docker HUB: <https://hub.docker.com/>

**1.1安装Centos7**







root账号密码为统一 123456

**1.2 安装Docker**

[root@node1 ~]# yum -y install docker #安装

[root@node1 ~]# systemctl start docker #启动Docker

[root@node1 ~]# systemctl enable docker #设置开机自启





**第三章 Docker基本操作**

Docker官网: <https://www.docker.com/>

Docker中文社区: <http://www.docker.org.cn/>

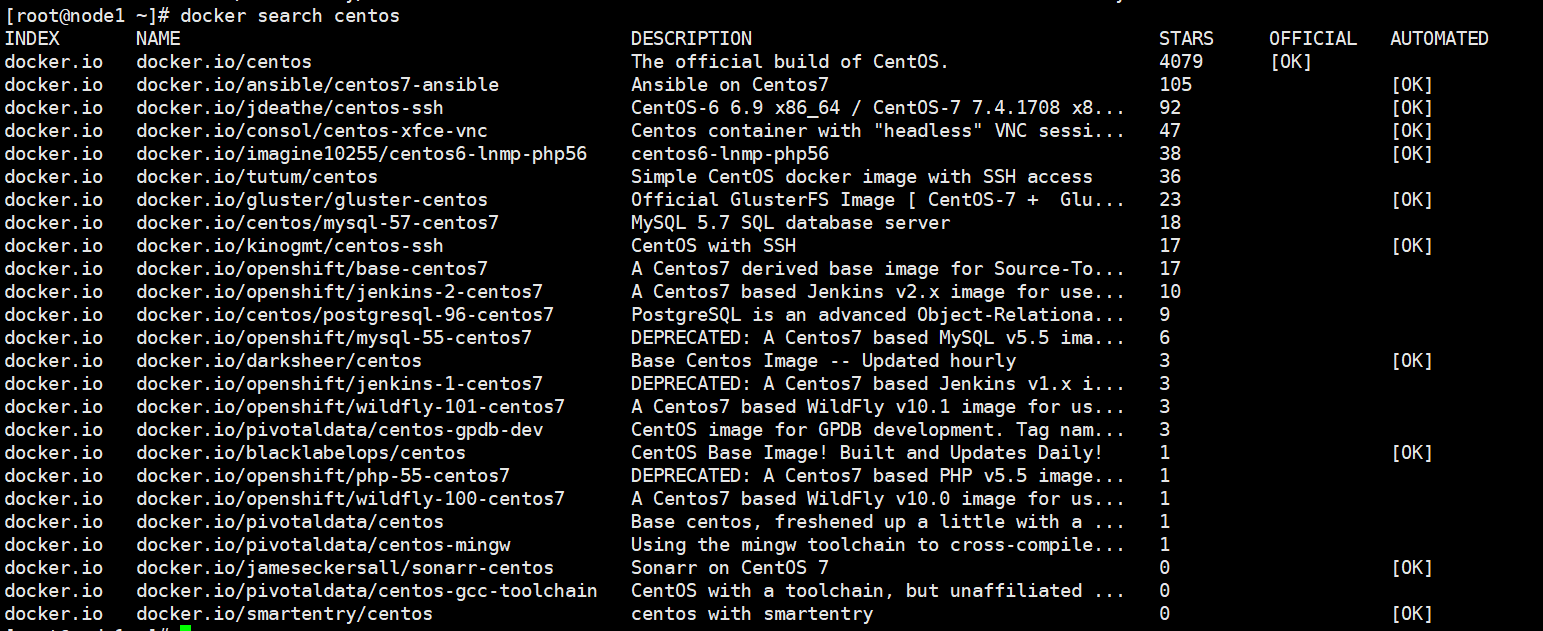
Docker HUB: <https://hub.docker.com/>

**docker <命令>**

**ps aux | grep docker docker 检查成功运行**

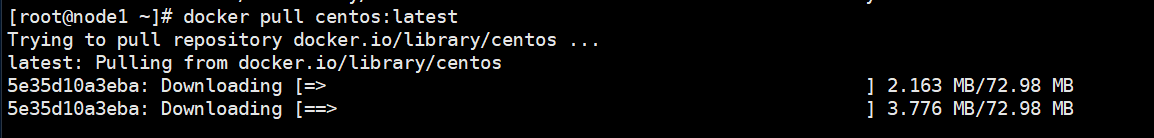
**search搜索镜像**

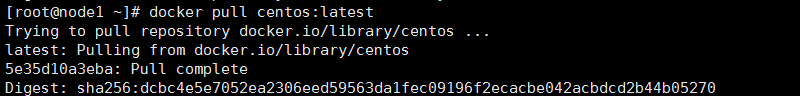
[root@node1 ~]# docker search centos



**pull下载镜像**

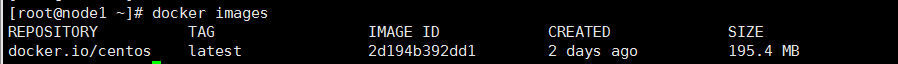
docker pull <image\_name>:<tag> latest 最新版 centos:7.2



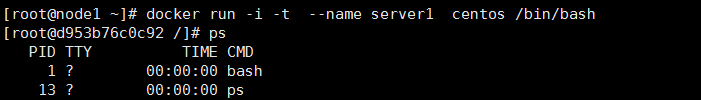


**images 列出镜像目录**

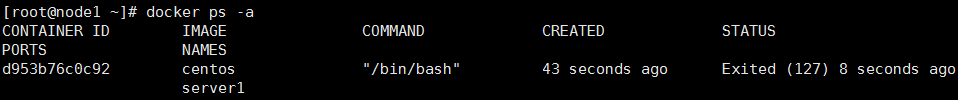
docker images



**run创建容器(-i -t 运行在bash下可以输入输出)**



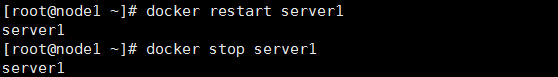
**ps查看容器列表(-a 标识未启动容器也显示)**



start 启动容器

restart 重启容器

stop 停止容器



attach连接容器

docker attach <容器名称/id>





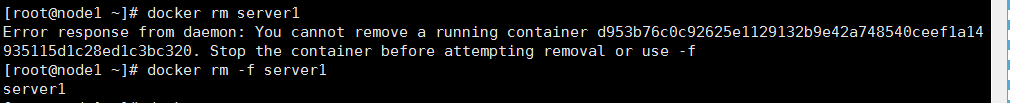
**exec外部运行容器内命令(容器中按照软件，不需要去容器内安装)**



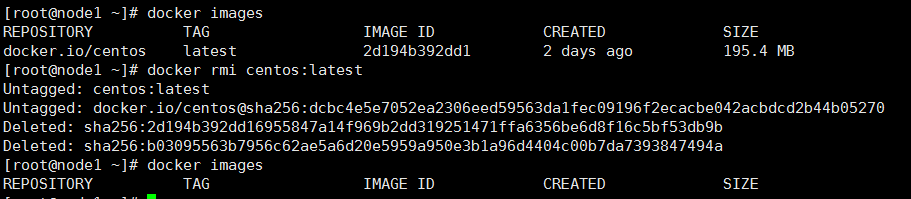
**rm 删除容器**

如果容器已停止 docker rm container\_name

如果容器还在运行 docker rm -f container\_name



**rmi 删除镜像**





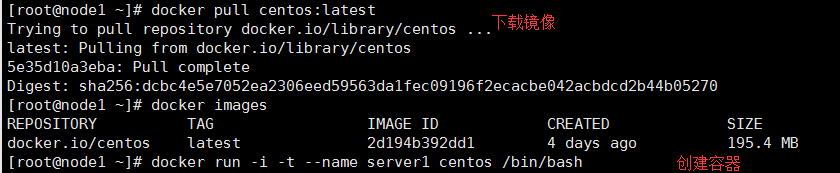
**第四章 Docker高级命令**

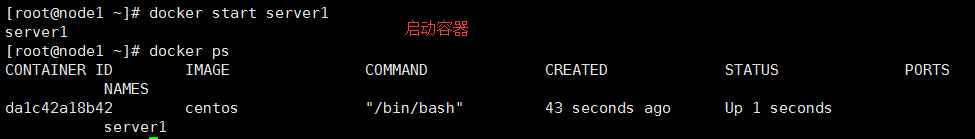
Docker官网: <https://www.docker.com/>

Docker中文社区: <http://www.docker.org.cn/>

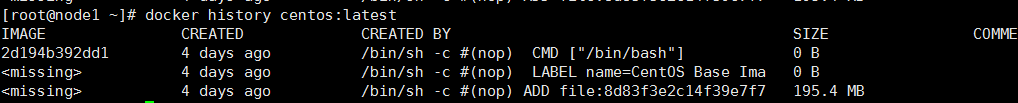
Docker HUB: <https://hub.docker.com/>

恢复试验环境



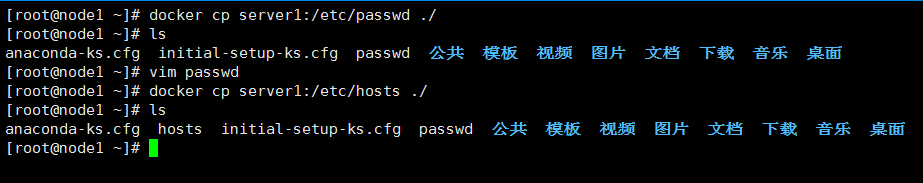


1. **使用history查看镜像历史**



1. **cp 命令复制文件**

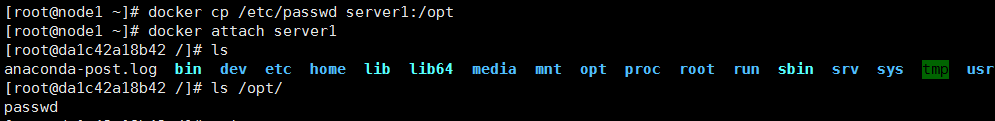
docker cp <容器名称>:<容器文件路径><本地主机的路径>



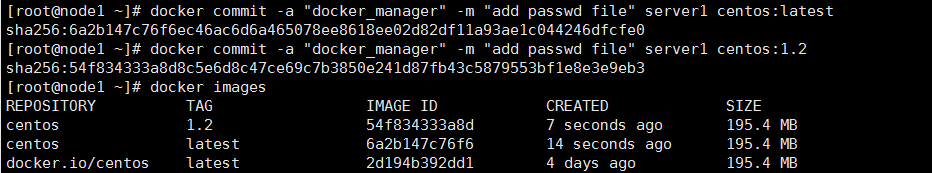
1. **commit命令提交容器修改**

docker commit <选项><容器名称><镜像名称>:<标签>

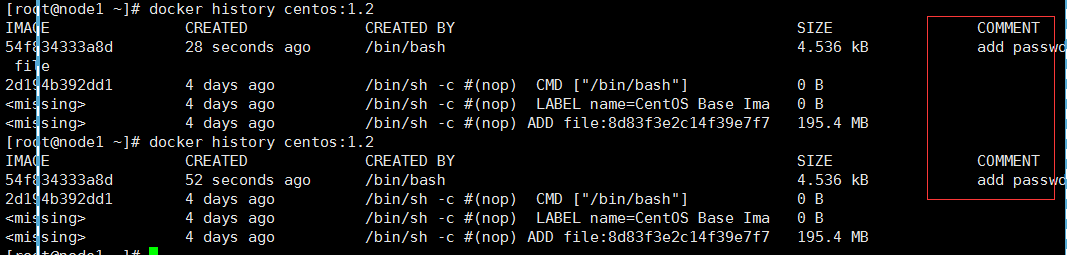
复制本地文件到容器中（我们对容器做了修改）



提交修改并查看本地镜像 -a(作者)

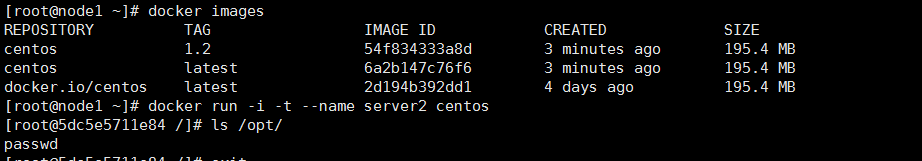


查看镜像历史

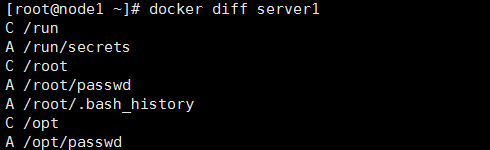


、

批量化部署应用的时候应该遵循这样的标准(加一层即为新发布一次，)



1. **diff命令检查容器文件的变动**

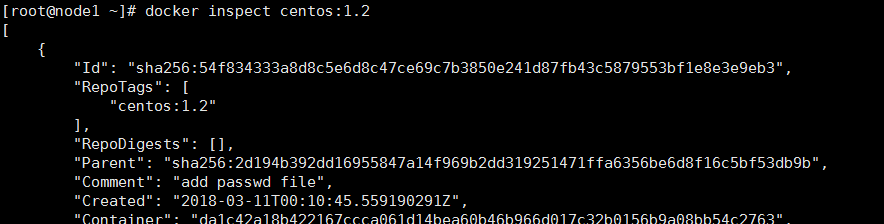


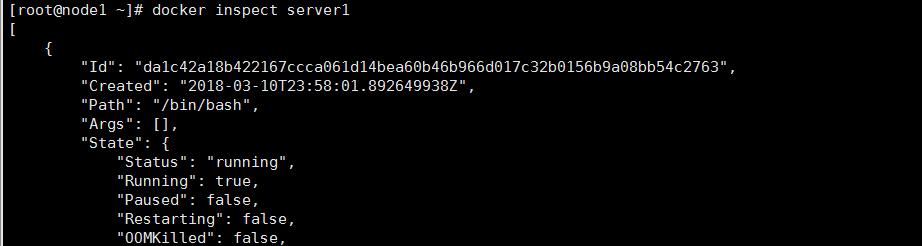
A：添加的文件

C：修改文件

D：删除文件

1. **inspect查看详细信息**







**第五章 Docker私有仓库**

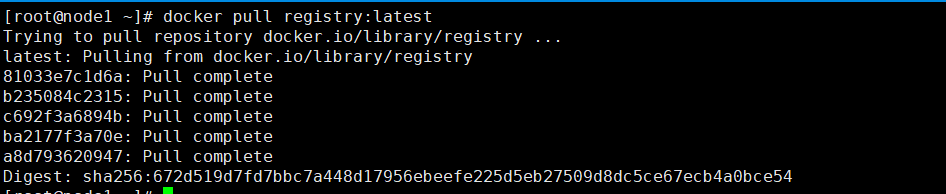
Docker官网: <https://www.docker.com/>

Docker中文社区: <http://www.docker.org.cn/>

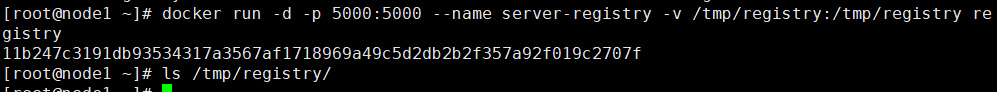
Docker HUB: <https://hub.docker.com/>

Docker push 将镜像上传到仓库

1. 下载Docker注册服务器镜像



1. 运行注册服务器

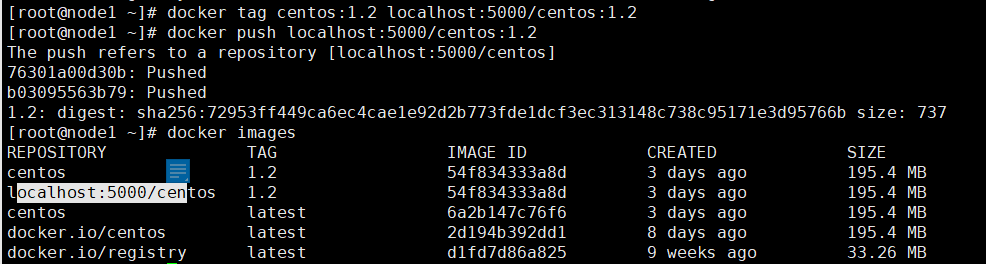


1. 上传镜像

向私有仓库上传镜像时，需要先创建标签。

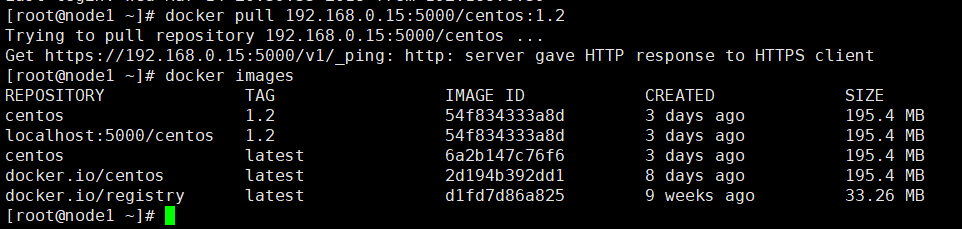
Docker tag <镜像名称>:<标记> <私有仓库地址>/<镜像名称>:<标记>

Docker push <私有仓库地址>/<镜像名称>:<标记>



1. 其他机器下载

Docker pull <私有仓库地址>/<镜像名称>:<标记>





**第六章 创建Docker镜像**

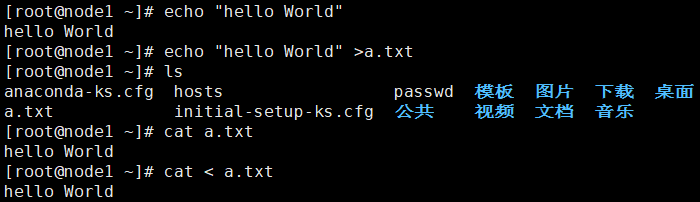
Docker官网: <https://www.docker.com/>

Docker中文社区: <http://www.docker.org.cn/>

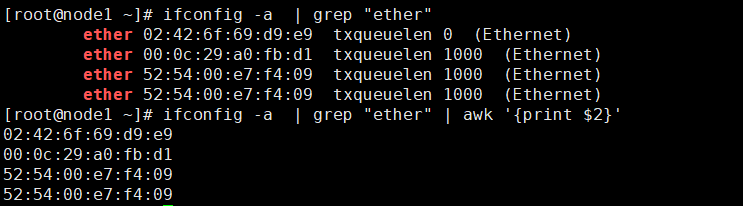
Docker HUB: <https://hub.docker.com/>

1. 熟悉Bash

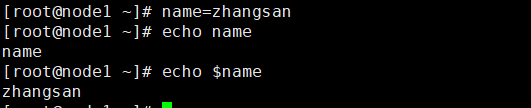
重定向输入和输出



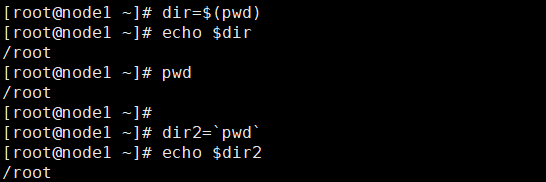
管道 命令连接



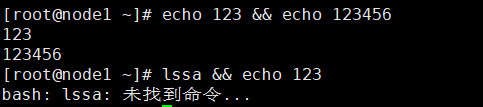
$ 引用变量



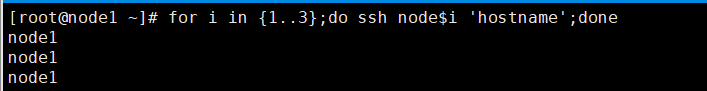
$() ``



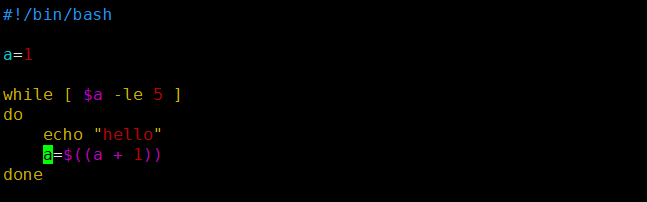
&&



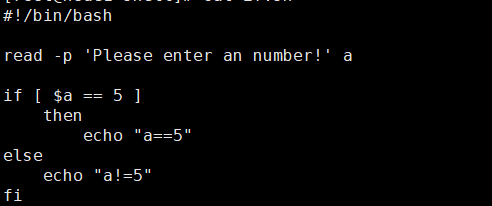
for循环



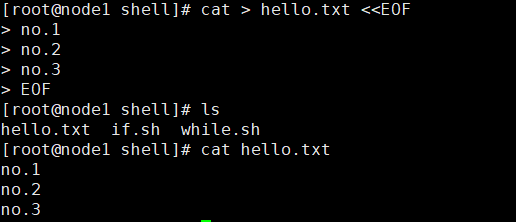
while循环



if语句



多行输入



1. 编写Dockerfile文件



VOLUME 与主机共享的目录

WORKDIR 工作目录

EXPOSE 指定端口

1. 使用Dockerbuild创建镜像

