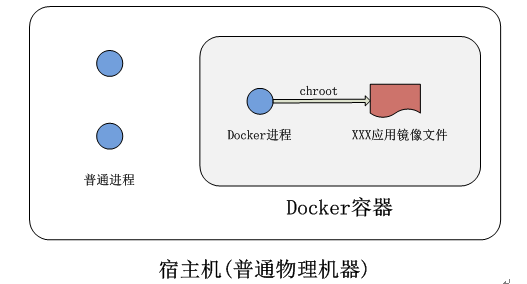
|  |
| --- |
| **想理解、会用Docker，这篇文章就够了！** |
|  |

**Docker是啥?**

打开翻译君输入Docker 结果显示码头工人，没错!码头工人搬运的是集装箱，那么今天要讲的Docker其操作的也是集装箱，这个集装箱就静态而言就是一个应用镜像文件，就动态而言，就是一个容器。蒙了吧?好吧，上图解释。



Docker从狭义上来讲就是一个进程，从广义上来讲是一个虚拟容器，其实更专业的叫法是应用容器( Application Container )，Docker进程和普通的进程没有任何区别，它就是一个普通的应用进程。不过是用来操作镜像文件的。所以Docker进程+构建的应用镜像文件就等于Docker容器。本文所有讲的Docker都是指Docker容器哦。

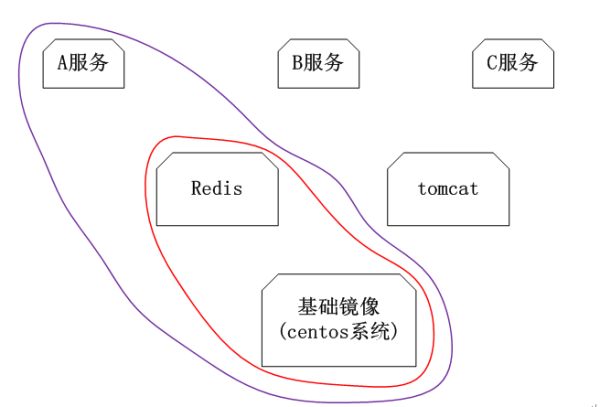
再继续下文之前我们首先要明确几个docker重要的基本概念吧，镜像，容器，仓库。

镜像Docker images，就类似于VM虚拟机里面的快照，但是可比快照轻量化多了。快照不懂?那可以把images直接理解成一个文件夹。我们可以通过ID或者易识别的名字+tag来确认唯一的目标镜像。ImagesID是一个64位的字符，但是一般我们都是使用前面12位就足够区别了。

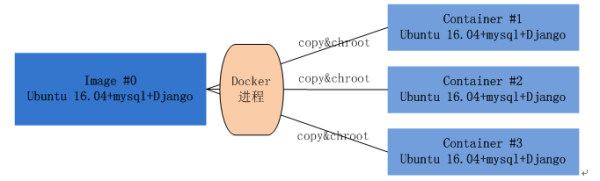
如图中左边红框中redis: lates和右边的红框中5f515359c7f8都唯一表示为同一个镜像。所以我们一般的镜像可以命名为类似centos:latest、centos:centos7.1.1503等等。

镜像是分层的，有基础镜像，仅仅包含操作系统，比如centos镜像;有中间件镜像，比如redis等数据库镜像;最后是应用镜像，就是指具体的应用服务了，应用镜像可以非常丰富，随时可以发布，这三者之间依次叠加。

所以当我们在使用 Docker构建镜像的时候，每一个命令都会在前一个命令的基础上形成一个新镜像层。如下图，基础镜像就是centos镜像，中间件镜像就是两个红色圈，应用镜像就是紫色圈。其中redis+centos这样叠加组合的中间件镜像就可以供A服务或者B服务使用，这样叠加组合更加灵活。仍和一种镜像都可以从Docker hub公共仓库中拉取。



容器Docker containers，你可以从镜像中创建容器，这如同从快照中创建虚拟机，不过更轻量，启动更快，秒启。应用是在容器中运行的，打个比方，你首先下载了一个Ubuntu的镜像，然后又安装mysql和Django应用及其依赖，来完成对它Ubutun镜像的修改，一个个人觉得非常完美应用镜像生成了!就把这个镜像分享给大家使用，大家通过这个镜像就生成一个容器。容器启动之后就会运行Django服务了。



上面也说到了，容器就是一个个独立的封闭的集装箱，但是也需要对外提供服务的，所以Docker允许公开容器的特定端口，在启动Docker的时候，我们就可以将容器的特定端口映射到宿主机上面的任意一个端口，所以，如果几个服务都需要80端口，那么容器的对外端口是80，但是映射到宿主机上面就是任意端口，就不会产生冲突，所以就不需要通过代理来解决冲突。容器对外端口与宿主机的端口映射可以通过下面的命令来完成。

**启动docker容器**

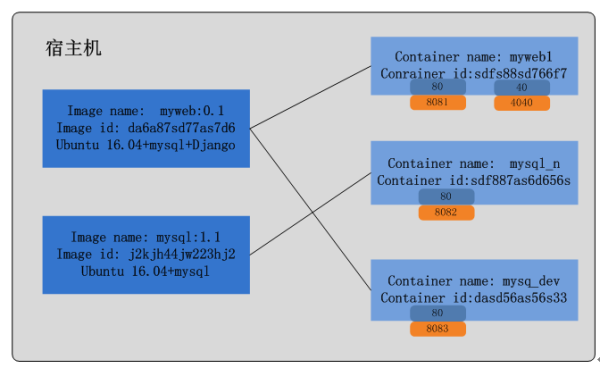
docker run -d -p 2222:22 --name 容器名 镜像名

-d 守护容器，就是后台运行，退出命令窗口容器也不会停止

-it 交互式容器 退出命令窗口容器就停止运行了

-p宿主机端口和容器端口映射

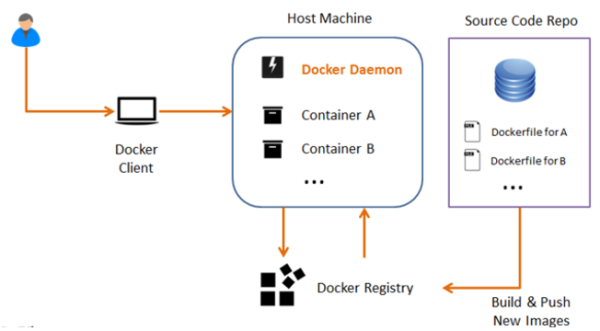
8081:80 宿主机端口:容器公开的端口



仓库Docker registeries，docker仓库和存放集装箱的仓库是一样的，不过docker使用来存放镜像的。仓库存在公有和私有之分，公有仓库docker hub提供了非常多的镜像文件，这些镜像直接拉取下来就可以运行了，你也可以上传自己的镜像到docker hub上面。同时也可以自己搭建私有仓库用于团队项目管理。

结合前面介绍的基本概念，我们可以将docker的几个概念使用大致串起来，他们之间是如何运作的，也就是Docker的生命周期。

看下图，主要是三步走。



开发构建镜像并将镜像push到Docker仓库

测试或者运维从Docker仓库拷贝一份镜像到本地

通过镜像文件开启Docker容器并提供服务

**为啥要用Docker?能干些啥?**

为啥要用Docker?这要从目前软件行业的痛点来讲起

软件更新发布及部署低效，过程繁琐且需要人工介入，

环境一致性难以保证，

不同环境之间迁移成本太高。

有了Docker可以很大程度解决上面的问题。

首先，Docker的使用简单至极，从开发的角度来看就是三步走：构建，运输，运行。其中关键步骤就是构建环节，即打包镜像文件。但是从测试和运维的角度来看，那就只有两步：复制，运行。有了这个镜像，那么想复制到哪运行都可以，完全和平台无关了。同时Docker这种容器技术隔离出了独立的运行空间，不会和其他应用争用系统资源了以及还不需要考虑应用之间相互影响，想想就开心。

其次，因为在构建镜像的时候就处理完了服务程序对于系统的所有依赖，所以在你使用的时候，你可以忽略掉原本程序的依赖以及开发语言。对测试和运维而言，更多专注于自己的业务内容上。

最后，Docker于开发者而言提供了一种开发环境的管理办法，与测试人员而言保证了环境的同步，于运维人员提供了可移植的标准化部署流程。

所以， Docker 能干啥，总结如下：

构建容易分发简单

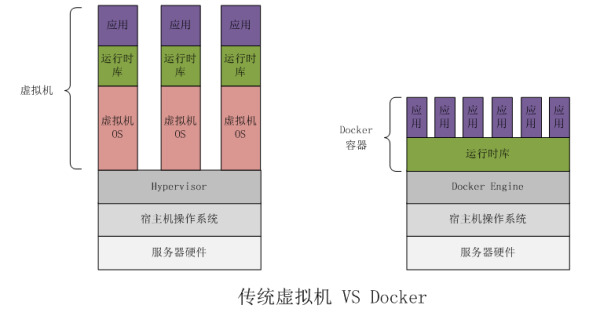
隔离应用解除依赖

快速部署测完就销

Docker是个进程级的轻量化虚拟机，和传统虚拟机有啥区别呢?

Docker这个虚拟机超级轻量级，仅仅是一个进程而已。与传统的虚拟机比如VM有着巨大的差别

区别看下图：



我们来看一下二者的区别，因为 VM 的 Hypervisor 需要实现对硬件的虚拟化，并且还要搭载自己的操作系统，其中虚拟机操作系统占用内存是比较大的，一个操作系统有好几个G，自然在启动速度和资源利用率以及性能上有非常大的开销，如果在本地，或者个人电脑，那么影响还不是那么大，但是在云端就是一个非常大的资源浪费。

咱们很多时候做事情的时候不会考虑与事情本身无关的问题，比如造飞机的不会考虑飞机是否要潜水，对于我们目前很多移动互联网的应用来说，很少会涉及到对操作系统的部分，其实我们主要关心的是应用的本身，而VM虚拟机的上层是运行的运行时库和应用，整个虚拟机的空间是非常的庞大，但是容器化技术Docker技术的出现后，省去了操作系统这一层，多个容器之间相互隔离且共用了宿主操作系统和运行时库。

所以Docker 应用容器相对于 VM 有以下几个优点：

启动速度快，容器启动本质就是一个开启一个进程而已，因此都是秒启，而 VM 通常要更久。

资源利用率高，一台普通 PC 可以跑成百上千个容器，你跑十个 VM 试试。

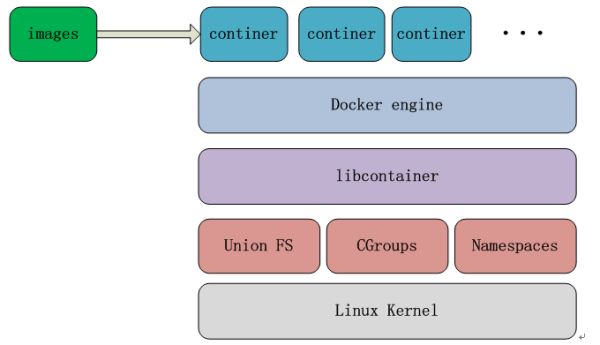
性能开销小， VM 通常需要额外的 CPU 和内存来完成 OS 的功能，这一部分占据了额外的资源。

所以很多移动互联网的应用或者云计算的后端节点都可以用docker来替换物理机器或者虚拟机。比如腾讯地图的很多后台服务基本上都迁移docker部署了。

**Docker是个啥架构?底层又是用的啥技术?**

前面说了那么多，始终还是雾里看花。下面就详细介绍一下技术架构，底层又是用的啥技术来实现上述那么多优点的?

Docker技术架构图：



从Docker依赖的底层技术来看，Docker原生态是不能直接在Windows平台上运行的，只支持linux系统，原因是Docker依赖linux kernel三项最基本的技术,namespaces充当隔离的第一级，是对Docker容器进行隔离，让容器拥有独立的hostname,ip,pid，同时确保一个容器中运行一个进程而且不能看到或影响容器外的其它进程;Cgroups是容器对使用的宿主机资源进行核算并限制的关键功能。

比如CPU,内存,磁盘等，union FS主要是对镜像也就是image这一块作支持，采用copy-on-write技术，让大家可以共用某一层，对于某些差异层的话就可以在差异的内存存储，Libcontainer是一个库，是对上面这三项技术做一个封装。

Docker engine 用来控制容器container的运行，以及镜像文件的拉取。

**Docker咋装呢?Docker怎么用呢?**

安装之前，我们首先确保自己的linux系统内核版本高于3.10，并且系统是64位，才能体验Docker哦。

通过uname -ir查看是否满足要求。

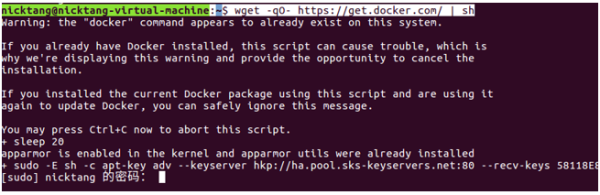
IMG_263

**Docker安装**

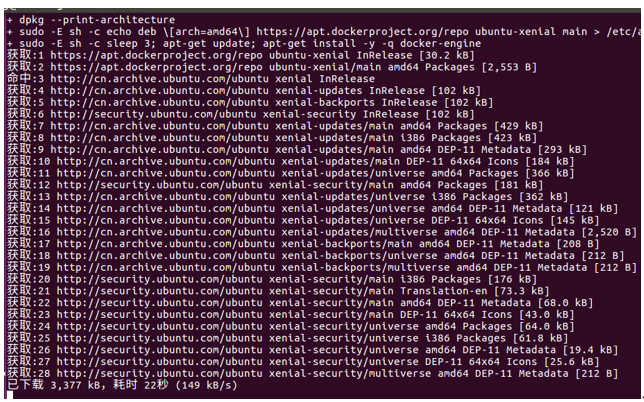
通过脚本的方式安装docker，非常简单。

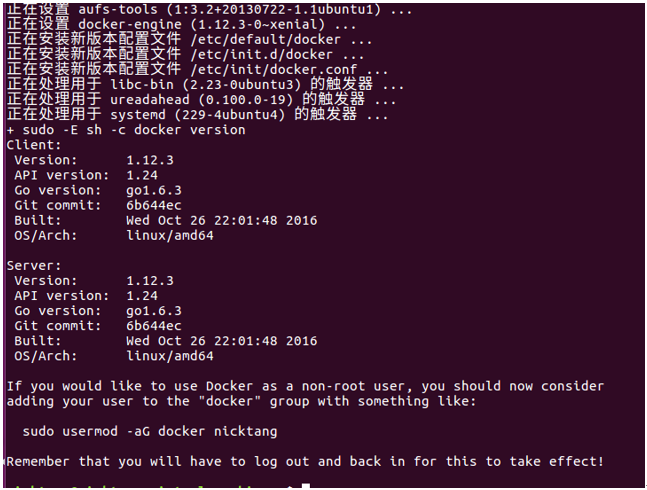
**1、 获取最新的docker安装包**

|  |
| --- |
| nicktang@nicktang -virtual-machine: ~$ wget -qO- https://get.docker.com/ | sh |



输入当前用户的密码后，就会下载脚本并且安装Docker及依赖包。





显示上图内容就表明安装完成。

**2、 启动docker 后台服务**

|  |
| --- |
| root@nicktang-virtual-machine: /data # sudo service docker start #启动守护进程  root@nicktang -virtual-machine:/data # docker -v |

IMG_267

能够看见版本号，说明docker的安装成功。简单吧!至此就差一个镜像了。自己制作还是从公共仓库拉取就随你啦。

|  |
| --- |
| root@nicktang-virtual-machine: /data # sudo service docker stop #关闭守护进程 |

**Docker使用**

Docker的使用，我们主要从【增删查】几方面来说说怎么使用docker,为什么没有【改】呢，因为在我看来docker容器一旦出现问题了，根本没有修复的必要，直接把容器停止并删除，再启动，这样来得快。所以我们只需要掌握几个基本命令即可，具体如下。

【查】查看本地已有的镜像 Docker images

IMG_268

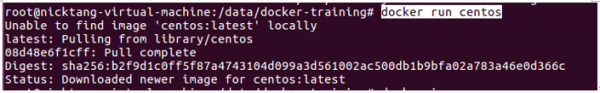
【增】运行一个镜像，即启动一个容器 docker run 镜像名 ，比如我们运行docker run centos

**键入这个命令的时候完成了三样操作**

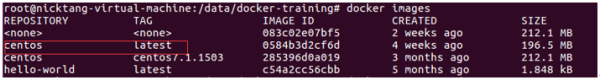
检查本地是否有hello-world这个镜像,有->就跳过第二步 没有->依次执行

就自动去docker hub下载这个镜像

就把镜像加载到容器并且运行



再用docker images查看的时候本地就增加了centos镜像。

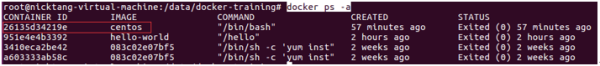


Tag为latest就表示是最新版本的centos系统镜像。因为会从docker hub拉取没有的镜像，所以算【增】里面。

【增】拉取指定的镜像文件 docker pull 镜像名:TAG

上面那种通过直接运行的方式拉取的是docker hub中最新的镜像，但是有时候我想拉取指定的镜像文件就需要使用docker pull命令来拉取。因为从官方拉取镜像文件，通常是比较慢的，所以我们可以通过加速器技术来从国内的镜像仓库拉取。

【查】查看所有的容器docker ps -a 可以用来查看所有的容器，包括运行中的和已经停止的。



第一个字段就是已经启动的容器ID，第二个字段就是这个容器是根据哪个镜像生成的。但是上面这个命令只是临时启动一下容器，上面图中的status 是exited(0),表示容器是退出状态。如果想容器在后台运行，所以我们需要启动守护式容器才可以，只要在启动命令中添加一个 -d参数，即docker run -d centos就可以了。

【查】查看镜像/容器的具体信息 docker inspect镜像ID(镜像名)/容器ID(容器名) docker inspect centos

这个命令是返回一个镜像或者容器详细信息的json串。其中包括ID，ip，版本，容器的主程序等非常多的信息，根据这些信息我们可以进行二次开发。在这个命令的基础之上增加一个-f参数我们可以指定获取自己需要的信息，比如获取redis容器的IP地址，内存信息，CPU使用情况。

|  |
| --- |
| docker inspect -f ‘{{.NetworkSettings.IPAddress}}’ [ID /Name] |

IMG_272

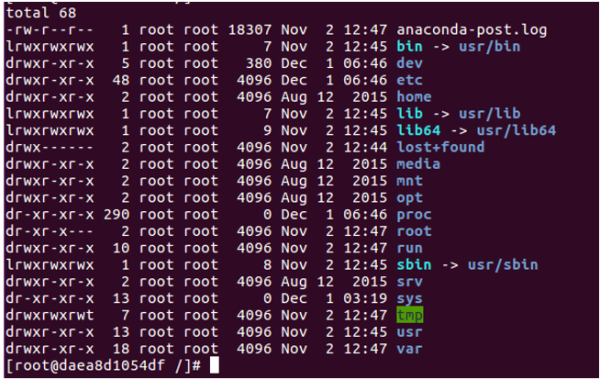
【查】进入容器 docker run -it centos 即启动一个交互式容器

-it 完成容器终端和当前终端进行关联，即当前终端的显示就会切换到容器终端的显示。

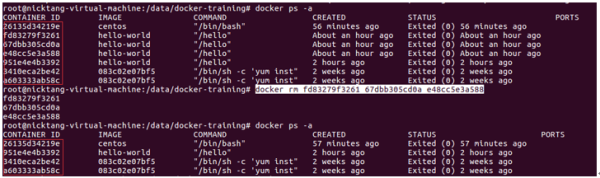
IMG_273

查看容器目录结构，发现和物理机器的目录结构完全一致，这就是为什么有的人称docker容器也称之为虚拟机的原因。

Exit可以退出容器终端。



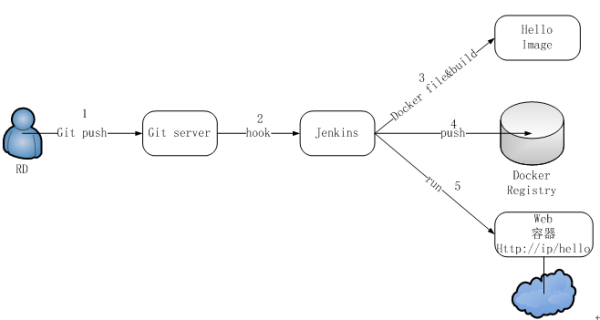
【删】删除容器，docker rm 容器ID，删除多个容器就可以多个容器ID之间用空格隔开即可。



**怎么用Docker完成持续集成、自动交付、自动部署?**

这年头见面不聊点自动化什么的，持续什么的，都不好意思。所以，咱们也要了解一下持续集成，自动交付，自动部署。但是上面说了这么多，没发现Docker有那三样功能啊，是的，Docker是没有这个功能，但是你在完成上述三样自动化的步骤都是依赖Docker的。Docker是这些流程实现的基础，就如同软件开发，软件代码才是根本，开发工具是辅助。所有搭建一个完整的自动化流程还需要github+jenkins+ registry三样帮助。

持续集成和自动部署的原理如下图所示：



RD推送代码到git 仓库或者svn等代码服务器上面，git服务器就会通过hook通知jenkins。

jenkine 克隆git代码到本地，并通过dockerFile文件进行编译 。

打包生成一个新版本的镜像并推送到仓库 ，删除当前容器 ，通过新版本镜像重新运行。

而在整个过程中 RD只需要敲入三个命令Git add \* ;Git commit –m “”;Git push即可完成持续集成、自动交付、自动部署。后面通过案例实际演示这个过程的神奇!

Docker还可以很方便的自动扩容哦，一般的自动扩容的两种方式，一种就是docker容量扩大，另一种就是docker节点数扩充。第一种就修改配置文件即可，第二种通过简单的拷贝，运行就完成了节点的扩容。

**总结**

虽然Docker具有超轻量化，但是不建议一台机器上面部署太多的应用，同时部署的时候一定要差异化部署，什么意思呢?就是将大量计算的，和内存需要大的，IO操作频繁的对系统资源需求不一致的部署到同一台宿主机上。