# 一、Linux 命令

命令 选项 参数

ls -a [文件]

## 查看路径

pwd ：显示当前所在路径

tree ：以树结果显示路径

## **查询与归档**

find -name “\*.txt”

压缩：tar cvf file.tar a.txt b.txt

解压缩：tar xvf file.tar

压缩gz包：tar -zvcf a.tar.gz a.txt b.txt

## **查看进程**

ps -A：显示所有进程

ps -u root：查看当前用户的进程

ps -ef | grep mysql：查看关于mysql的进程

## **查看文件**

cat a.txt 显示整个文件内容

head a.txt

head -n 2 a.txt：查看文件前2行

head -n -2 a.txt：过滤文件后2行，只看前面的

## **排序和过滤**

sort a.txt : 对文件内的内容排序，以AcsLL码排序

sort -u a.txt: 去重

sort -r a.txt : 降序排序

grep——文本搜索

ls -l : 查看所有文件目录

ls -l | grep txt : 从所有件目录中搜索txt文件名

ps aux | grep ssh :从所有进程中搜索ssh进程

grep com a.txt :从文件中找特定字符“com”[可使用正则表达式]

sed——编辑处理文本工具

sed ‘s/a1/a2’ a.txt :将文件中的a1替换成a2

## **目录处理--建立/删除/复制/粘贴**

mkdir log01 ：在本路径下建立文件目录

mkdir -p janpan/log ：一长串目录建立

cd /root/log ：切换目录

cd .. ：切到上级目录

touch log01：建立空目录

【及其少见】rmdir log01/ :是删除空目录，如果目录下有文件，则删不掉

rm -rf [文件/目录] ：强制删除任何的文件或目录

rm -rf janpan/log ：强制删除目录，不会问是否删除\*\*文件和目录

如果用rm -r janpan/log：会问你是否删除log？ 是否删除janpan? 每次打Y很麻烦，所以一次性用强制性删除所有

cp -r janpan/ /tmp/ ：复制目录到tmp/下

cp abd.txt /tmp/nam : 将文件复制到tmp/下，并重新命名为nam

cp -a abd.txt /tmp/ : 将文件复制到tmp/下,与源文件属性一致，包括创建的时间

剪切或改名

mv janpan/ /tmp/janpan1 : 将目录移到目标目录

mv log.txt abc.txt : 原文件和目标文件在同一目录，则是改名字

## **目录作用**

根目录下的bin 和 sbin ；usr目录下的bin 和 sbin ——用于保存系统命令，sbin的命令只有root权限、超级权限才能使用的命令

boot ——系统启动数据，不能随便操作

etc ——存储配置文件

lib——函数库,用的时候调用

media mnt misc ——三个空目录，作为外接设备挂载目录；U盘

var——保存可见文档

root/home/tmp : 三个目录可以任意放东西，其他文件不要动

## **日志文件**

查看日志文件（默认情况下 tail 命令只会显示文件最后 10 行的内）

tail -f /var/log/kern.log

如果在文件的路径里，tail -f +文件名即可

log文件存放在/var/log/目录下，该目录是系统日志文件的保存位置

## **Vim编辑**

(1)按i，进入编辑状态；

(2)按ESC退出编辑模式后

同时按shift+:

左下角可以进行输入~

:w 保存但不退出

:wq 保存退出

:q 退出

## **查看磁盘**

Linux磁盘管理常用三个命令为df、du和fdisk。

## **linux两台机器之间传文件**

1. scp传输
2. grep管道传输

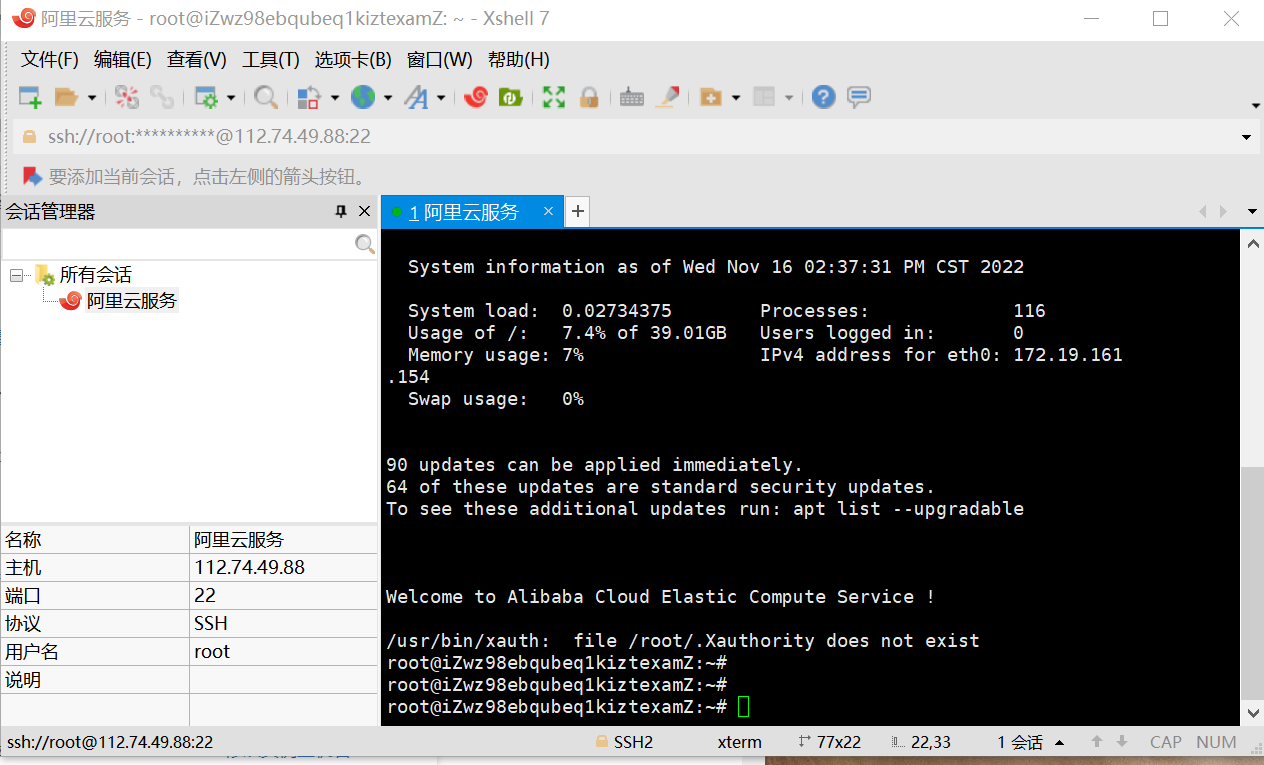
# linux系统

方法1：安装虚拟机，在虚拟机里安装linux

方法2：购买/免费试用云服务器

阿里云服务账户密码：





# Docker

## **1.docker框架**

类似于轻量级虚拟机：提供可以独立运行应用的环境

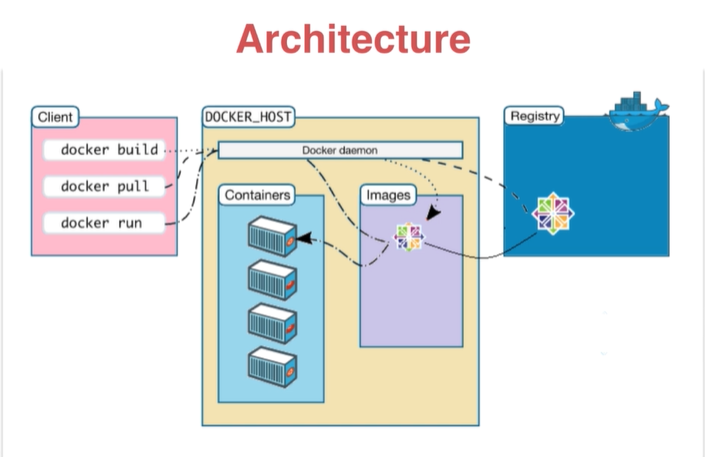
Clinet：就是命令行的操作界面，通过其使用Api 接口，完成和Doceker daemon 的通信

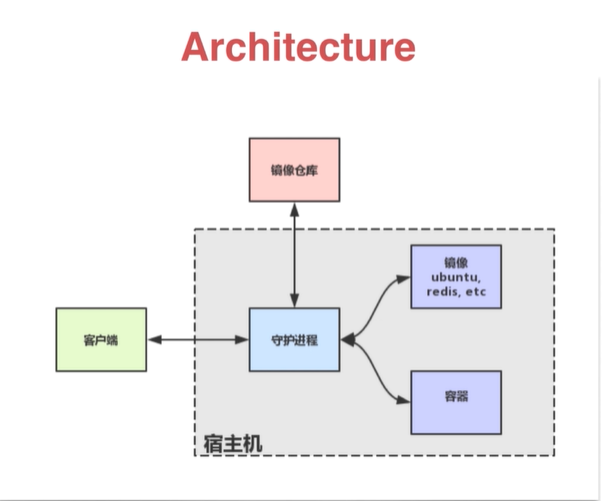
Docker\_Host: Docker 的宿主机，可以是物理机，也可以是虚拟机，上面跑着一个dockers 服务，我们使用 docker daemon 实现对docker 对象生命周期管理。

Registey: 镜像仓库，是docker 镜像的存储中心，docker 的官方仓库为docker hub。为了方便在开发团队中我们会维护一个仓库，这个仓库我们一般使用 harbor 实现—私有仓库。

Daemon:守护程序，与客户端命令交互

Docker :在我们安装服务的时候避免对宿主机繁琐的配置,就可以将一个软件运行起来。以前我们开发一个java程序,宿主机必须提供jre 环境。如果没有这个环境,不好意思该程序无法执行,有了容器后这些就简单了。拉取一个镜像,然后运行起来,结束。





## **Docker的应用场景与优点**

* Web 应用的自动化打包和发布
* 自动化测试和持续集成、发布

优点：

Docker 允许开发人员使用您提供的应用程序或服务的本地容器在标准化环境中工作，从而简化了开发的生命周期。

容器非常适合持续集成和持续交付（CI / CD）工作流程

## **构建镜像方法**

将自己写的Java程序制作成镜像，运行起来，生成容器，不需要额外去配置jre环境

Dockerfile 是一个用来构建镜像的文本文件，文本内容包含了一条条构建镜像所需的指令和说明.

1.创建Dockerfile文件

（1）mkdir d1

（2）cd d1【在d1路径下创建dockerfile】

（3）touch Dockerfile

（4）vim Dockerfile

#From aipine:latest

MAINTAINER xbf

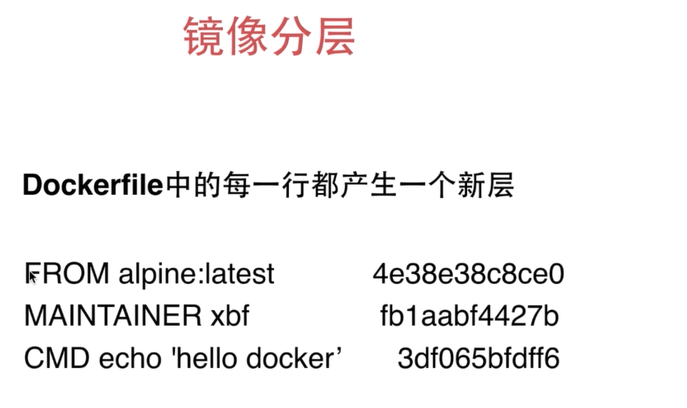
CMD echo “hello Docker”#

（5）docker build -t hello\_docker . 【点表示该路径下所有的东西转成镜像，该路径下只有 1个dockerfile】

（6）docker images hello\_docker

（7）docker run hello\_docker 运行镜像

（7）docker commit 上传镜像（镜像仓库）



镜像是共享的，分层的内容有相同

docker file 类似于bash 文件

FROM: base image

Run: 执行命令

ADD：添加文件

COPY：拷贝文件

CMD：执行命令

2.[docker](https://so.csdn.net/so/search?q=docker&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_39621075/article/details/_blank) commit 命令

[docker](https://so.csdn.net/so/search?q=docker&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_39621075/article/details/_blank) commit mycentos7 centos7my #docker commit 容器名 新的镜像名

## **镜像与容器区别**

Docker 镜像可以看作是一个特殊的文件系统，静态的，镜像文件就是dockerfile，将应用程序与该程序的依赖，打包在一个文件里面，像数据库等；运行这个文件，就会生成一个虚拟容器，就可以让程序跑起来，从而不用再去考虑环境问题。

镜像是静态的，容器是动态的，镜像与容器为一对多的关系，容器可以被创建、启动、停止、删除、暂停。

一个镜像怎么运行为容器呢？比如msql，我们下载一个mysql镜像

docker pull mysql

创建mysql容器：运行镜像，生成mysql\_zhangsan数据库(专门给张三用的数据库)

docker run ‐p 3306:3306 ‐‐name mysql\_zhangsan ‐e MYSQL\_ROOT\_PASSWORD=123456 ‐d  mysql

-p 3306:3306 : 端口映射[宿主机的端口:容器的端口]

--name mysql\_zhangsan : 给这个容器命名为mysql\_zhangsan（这个名字是唯一的，不能重复）

-e MYSQL\_ROOT\_PASSWORD=123456 : 这个数据库登陆的账号为ROOT 密码为123456

-d 是基于我们刚刚拉下拉的镜像mysql进行创建的。

运行镜像，生成mysql\_lisi数据库(专门给李四用的数据库)

docker run ‐p 3307:3307 ‐‐name mysql\_lisi ‐e MYSQL\_ROOT\_PASSWORD=123456 ‐d  mysql

-p 3307:3307 : 端口映射（不能用3306了，因为3306被张三的数据库占用了）

--name mysql\_lisi : 给这个容器命名为mysql\_lisi（这个名字是唯一的，不能和上面的mysql\_zhangsan重复）

-e MYSQL\_ROOT\_PASSWORD=123456 : 这个数据库登陆的账号也为ROOT 密码也为123456

-d 是基于我们刚刚拉下拉的镜像mysql进行创建的。

到这里，我们成功通过一个镜像，创建了两个不同的容器。这样，我们的电脑上就能运行两个mysql了。

## **[Docker](https://so.csdn.net/so/search?q=Docker&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_33286909/article/details/_blank)镜像部署**

### 1.Doker部署jenkins

安装前准备  
1:jenkins镜像内部默认用户ID为1000  
2:jenkins镜像内部默认主目录为/var/jenkins\_home  
3:个人自定义暂定为/data/besttop/jenkins（可根据自己的目录做修改）

docker pull [jenkins](https://so.csdn.net/so/search?q=jenkins&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_33286909/article/details/_blank)/jenkins # 拉取镜像

mkdir -p /data/besttop/jenkins 创建个人目录

chown 1000:1000 /data/besttop/jenkins

docker run -d --name jenkins-prod -p 20031:8080 -p 50000:50000 -v /data/besttop/jenkins:/var/jenkins\_home jenkins/jenkins:lts【宿主机与容器目录的映射，方便在宿主机更好的查看容器文件】（使用-v挂载目录，冒号前面部分是主机要挂载文件路径，冒号后面部分是挂载到容器的路径）

最后：访问 <http://主机ip:20031/>

### **2.Docker部署代码**

* 开发人员每天把代码提交到 Gitlab 代码仓库
* Jenkins 从 Gitlab 代码仓库中拉取项目源码到Jenkins的工作目录，进行编译并打成 jar 包（命令: mvn clean package）
* 在构建后的步骤中：设置生产服务器IP以及 SSH 远程命令【命令：构建镜像，将镜像推送到生产服务器的私有镜像仓库，让生产服务器从镜像仓库中拉取镜像到本地，然后创建容器】

构建镜像push到指定服务器上：发布到生产服务器上运行

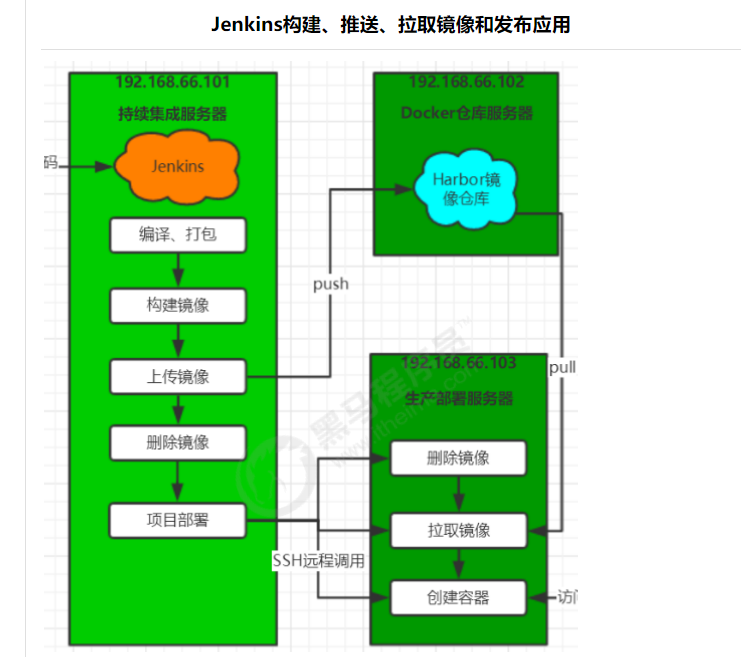
docker build -t 192.168.1.121:5000/hxkg-tomcat-8.5 .

docker push 192.168.1.121:5000/hxkg-tomcat-8.5

* 最后用户可以访问到容器

如果不用镜像，就是在build处用命令打包在jenkins目录下的源码，然后在构建后的步骤中：直接复制到生产服务器中并运行文件【shell命令】

注意：别人设置为私有的镜像，外人直接拉取不了，得让镜像管理人给你开通权限



## **Docker操作命令**

进入docker里面才能使用

镜像操作：

docker pull :获取image

docker build : 创建image

docker images：列出images

docker rmi [image id]：删除镜像

docker commit : 保存改动为新的image

容器操作：

docker run [容器id/镜像名]: 运行容器

docker stop [容器id] ：停止容器

docker ps ：查看正在运行的容器

docker ps -a : 查看所有的容器

docker rm [容器id]: 删除容器

docker cp : 在host 和容器之间拷贝文件

docker exec -it mysql /bin/bash : 进入容器，操作mysql

构建镜像push到指定服务器上：发布到生产服务器上运行

docker build -t 192.168.1.121:5000/hxkg-tomcat-8.5 .

docker push 192.168.1.121:5000/hxkg-tomcat-8.5

## **Docker compose**

比如有些 APP 由多个容器组成（php，mysql）等等，要创建、管理这些 [container](https://so.csdn.net/so/search?q=container&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_43972437/article/details/_blank)（启动停止删除），部署的时候会非常的繁琐，所以使用 docker-compose 来管理这些容器，主要是通过一个 yml 文件来定义。

1、编写 docker-compose.yml 文件

  
2、命令行管理 docker-compose（启动停止等）

