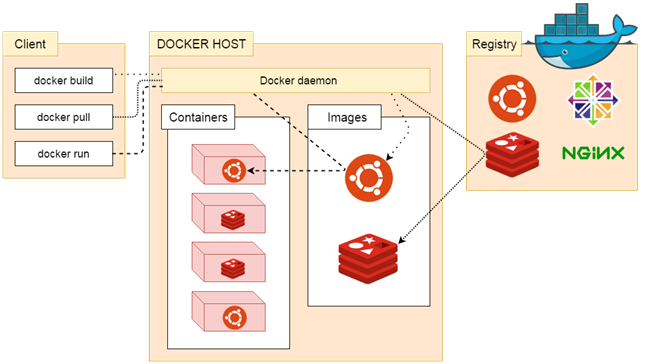
# 基本概念

1. <https://yeasy.gitbook.io/docker_practice/basic_concept/image>

2. <https://www.cnblogs.com/sparkdev/p/9092082.html>

3. <https://www.yiibai.com/docker/docker-architecture.html>

## 零、Docker基本结构：



## 一、镜像：

我们都知道，操作系统分为内核和用户空间。对于Linux而言，内核启动后，会挂载root文件系统为其提供用户空间支持。而Docker镜像（Image），就相当于是一个root文件系统。比如官方镜像Ubuntu:18.04就包含了完整的一套Ubuntu 18.04最小系统的root文件系统。

Docker镜像是一个特殊的文件系统，除了提供容器运行时所需的程序、库、资源、配置等文件外，还包含了一些为运行时准备的一些配置参数（如匿名卷、环境变量、用户等）。镜像不包含任何动态数据，其内容在构建之后也不会改变。

## 二、分层存储：

因为镜像包含操作系统完整的root文件系统，其体积往往是庞大的，因此在Docker设计时，就充分利用了Union FS的技术，将其设计为分层存储的架构。所以严格来说，镜像并非是像一个ISO那样的打包文件，镜像只是一个虚拟的概念，其实际体现并非由一个文件组成，而是由一组文件系统组成，或者说，由多层文件系统联合组成。

镜像构建时，会一层层构建，前一层是后一层的基础。每一层构建完就不会再发生改变，后一层上的任何改变只发生在自己这一层。比如，删除前一层文件的操作，实际不是真的删除前一层的文件，而是仅在当前层标记该文件已删除。在最终容器运行的时候，虽然不会看到这个文件，但是实际上该文件会一直跟随着镜像。因此，在构建镜像的时候，需要额外小心，每一层尽量只包含该层需要添加的东西，任何额外的东西应该在该层构建结束前清理掉。

分层存储的特征还使得镜像的复用、定制变的更为容易。甚至可以用之前构建好的镜像作为基础层，然后进一步添加新的层，以定制自己需要的内容，构建新的镜像。

## 三、容器：

## 四、仓库：

## 五、守护进程：

守护进程Daemon，这是一个监听Docker API请求的进程，它还管理Docker对象，如：映像，容器，网络等等。

守护进程还可以与其它守护进程通信以管理Docker服务。

每一个容器的沙箱隔离特性就是由守护进程来实现的。

## 六、Docker客户端：

Docker命令行工具，是用户使用Docker的主要方式，Docker Client与Docker Daemon通信并将结果返回给用户，Docker Client也可以通过Socket或者RESTful Api访问远程的Docker Daemon。