1. 爬虫的架构
2. 新浪的登录流程

登录前会有prelogin来获取timestamp，nonce，pubkey,servertime,rsakv

1. http和https的区别

https在TCP层和应用层中间加了一个SocketSecureLayer或TransportSecureLayer，在TCP三次握手后，再进行SSL握手，首先客户端给服务器发送自己的浏览器版本，支持加密格式，以及一个随机数。然后服务器给客户端发送一个CA（Certification Authority数字证书）证书以及一个随机数。客户端通过这个CA证书来获取服务器的公钥，然后通过这个公钥加密一个对称秘钥（由三个随机数生成）给服务器。然后服务器用自己的私钥来解密私钥。最后整个会话期间全部用对称的秘钥加密。

此密钥有时间限制，过期会被服务器删掉。秘钥前两个随机数是可以被监听的。所以破解秘钥最关键的是第三个随机数

若ssl中断需要重新连接，但如果有sessionID或者sessionticket的话就不需要重新生成秘钥了

http://www.mahaixiang.cn/internet/1233.html

1. session 与 cookie的区别

cookie为保存在本地的数据

session为保存在服务器端的记录，基于cookie实现

cookie存储以下信息

Set-Cookie: name = VALUE;

expires = DATE;

path = PATH;

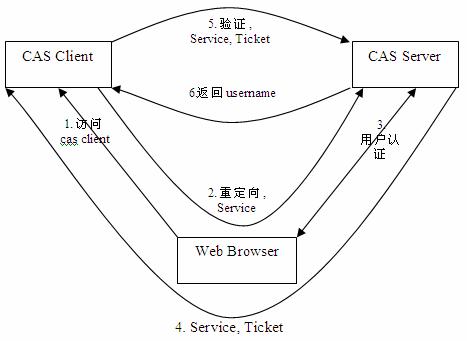
domain = DOMAIN\_NAME;

而session通过在本地保存一个存有sessionid 的cookie，发送给服务器来识别

1. 微博的sso（single sign on）实现了yale-cas

CAS 原理和协议

从结构上看，CAS 包含两个部分： CAS Server 和 CAS Client。CAS Server 需要独立部署，主要负责对用户的认证工作；CAS Client 负责处理对客户端受保护资源的访问请求，需要登录时，重定向到 CAS Server。图1 是 CAS 最基本的协议过程：



CAS 基础协议

CAS Client 与受保护的客户端应用部署在一起，以 Filter 方式保护受保护的资源。对于访问受保护资源的每个 Web 请求，CAS Client 会分析该请求的 Http 请求中是否包含 Service Ticket，如果没有，则说明当前用户尚未登录，于是将请求重定向到指定好的 CAS Server 登录地址，并传递 Service （也就是要访问的目的资源地址），以便登录成功过后转回该地址。用户在第 3 步中输入认证信息，如果登录成功，CAS Server 随机产生一个相当长度、唯一、不可伪造的 Service Ticket，并缓存以待将来验证，之后系统自动重定向到 Service 所在地址，并为客户端浏览器设置一个 Ticket Granted Cookie（TGC），CAS Client 在拿到 Service 和新产生的 Ticket 过后，在第 5，6 步中与 CAS Server 进行身份合适，以确保 Service Ticket 的合法性。

在该协议中，所有与 CAS 的交互均采用 SSL 协议，确保，ST 和 TGC 的安全性。协议工作过程中会有 2 次重定向的过程，但是 CAS Client 与 CAS Server 之间进行 Ticket 验证的过程对于用户是透明的。

另外，CAS 协议中还提供了 Proxy （代理）模式，以适应更加高级、复杂的应用场景，具体介绍可以参考 CAS 官方网站上的相关文档。

1. http的请求格式

<http://blog.csdn.net/hudashi/article/details/50789006>

HTTP 请求由三部分组成：请求行、 请求头和请求正文。

(1)、请求行：请求方法 URI 协议/版本

(2)、请求头(Request Header)

(3)、 请求正文

下面是一个HTTP请求的数据：

POST /index.php HTTP/1.1

Host: localhost

User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:10.0.2) Gecko/20100101 Firefox/10.0.2

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: zh-cn,zh;q=0.5

Accept-Encoding: gzip, deflate

Connection: keep-alive

Referer: <a target=\_blank href="http://localhost/" style="color: rgb(51, 102, 153); text-decoration: none;">http://localhost/</a>

Content-Length：25

Content-Type：application/x-www-form-urlencoded

username=aa&password=1234

请求的第一行是“方法 URL 协议/版本”，并以 回车换行作为结尾。请求行以空格分隔。格式如下：

POST /index.php HTTP/1.1

以上代码中“POST”代表请求方法，“/index.php”表示URI，“HTTP/1.1代表协议和协议的版本。

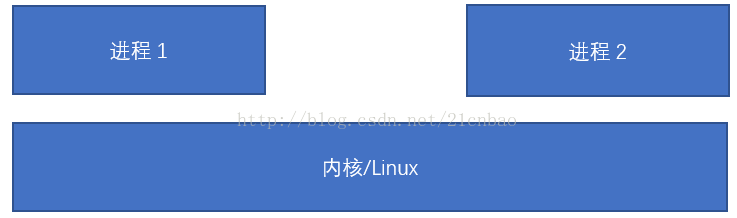
根据HTTP标准，HTTP请求可以使用多种请求方法。例如：HTTP1.1支持7种请求方法：GET、POST、HEAD、OPTIONS、PUT、DELETE和TARCE。在Internet应用中，最常用的方法是GET和POST。

URL完整地指定了要访问的网络资源，通常只要给出相对于服务器的根目录的相对目录即可，因此总是以“/”开头，最后，协议版本声明了通信过程中使用HTTP的版本。

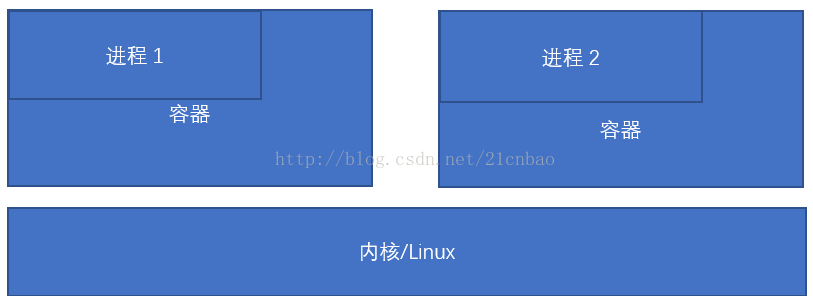
1. Docker

Blog： <http://blog.csdn.net/21cnbao/article/details/56275456>

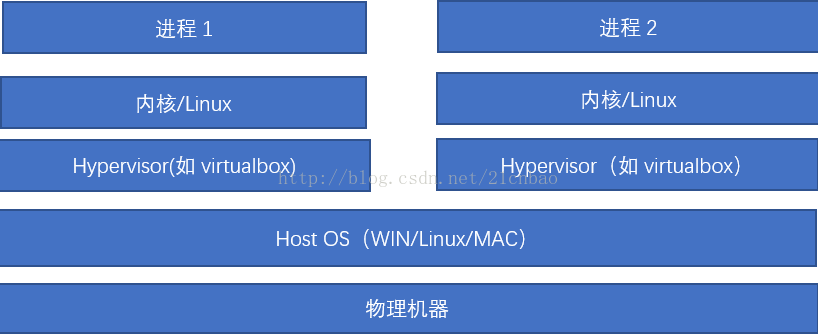
KVM, Virtualbox, Vmware是虚拟出机器，让每个实例看到一个单独的机器；而Docker是虚拟出操作系统，实现应用之间的隔离，让各个应用觉得自己有一个自己的操作系统，而且彼此之间隔离。假设没有Docker，然后有进程1和进程2，它们的运行将类似下图，进程1和进程2共享kernel，它们是同一OS下2个进程，因此必须拥有不同PID，但是又共享网卡，共享IP地址，看到一样的根文件系统（不chroot的情况下）等，可以用Linux IPC手段进程间通信。



有Docker的情况下，假设进程1和进程2运行于不同的容器，那么进程1和进程2都觉得自己和对方没有半毛钱关系，都觉得自己拥有自己的根文件系统，自己的网卡等，然后进程1和进程2的PID还可以一样，比如假设2个都是100。但是，此100非彼100。

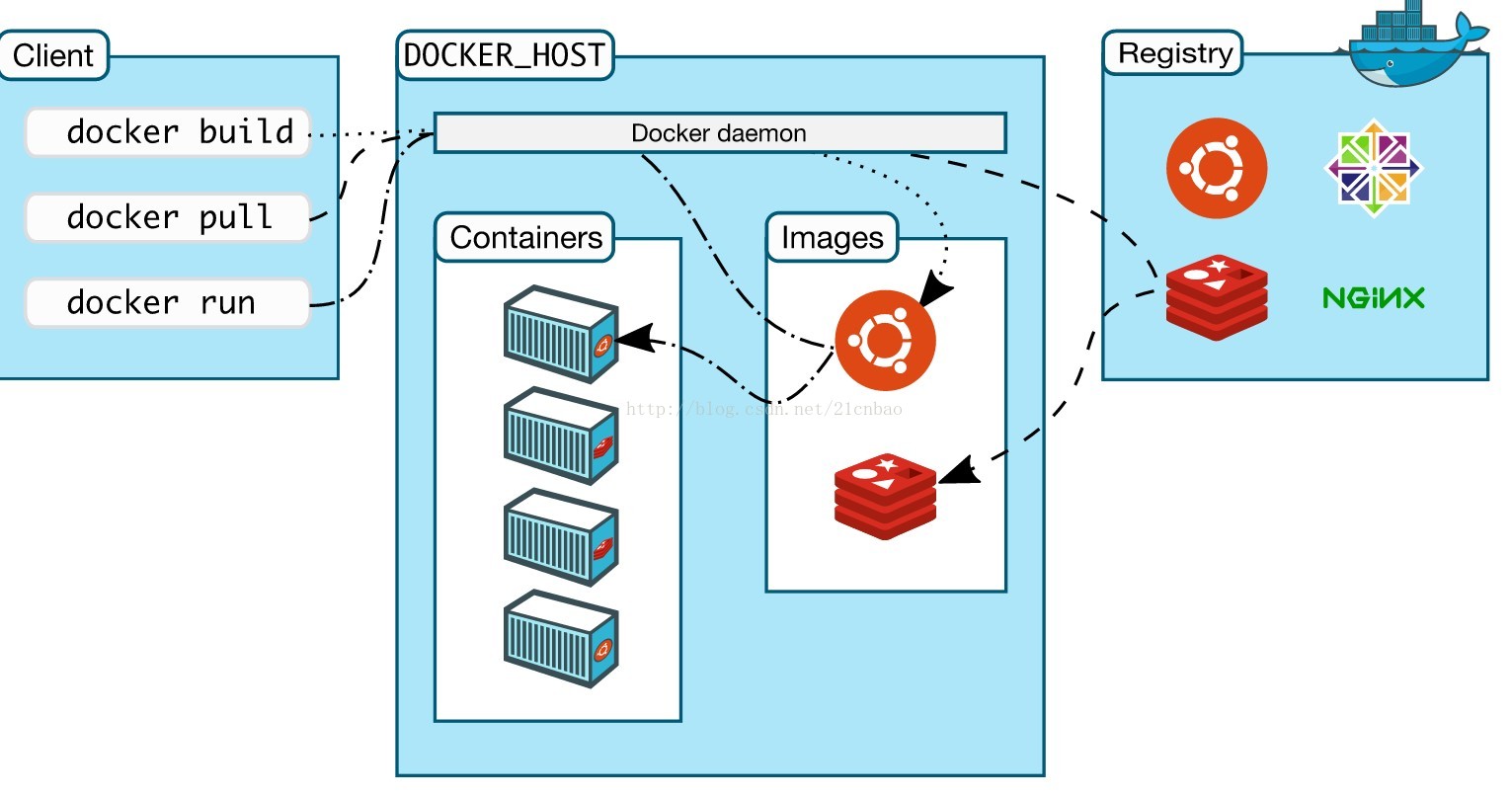


Virtualbox等虚拟机的思路则完全不一样，如果进程1和进程2运行于不同的虚拟机，则操作系统都是双份的，它们感觉自己在不同的虚拟电脑上面跑。



由于可见，Docker达到了类似虚拟机的效果，但是又没有虚拟机的开销，它虚拟的层次更加高。Docker不虚拟机器，仅仅虚拟应用的运行环境。

Docker中可能涉及到3个机器或者更多机器，一个运行docker命令的client， 一个包含images并以容器(container)形式运行image的主机，一个docker的images仓库。client与docker host上面的docker daemon通信。当然docker client和host可以运行于一台机器（我们做实验的时候是一台），默认的docker仓库是Docker Hub。

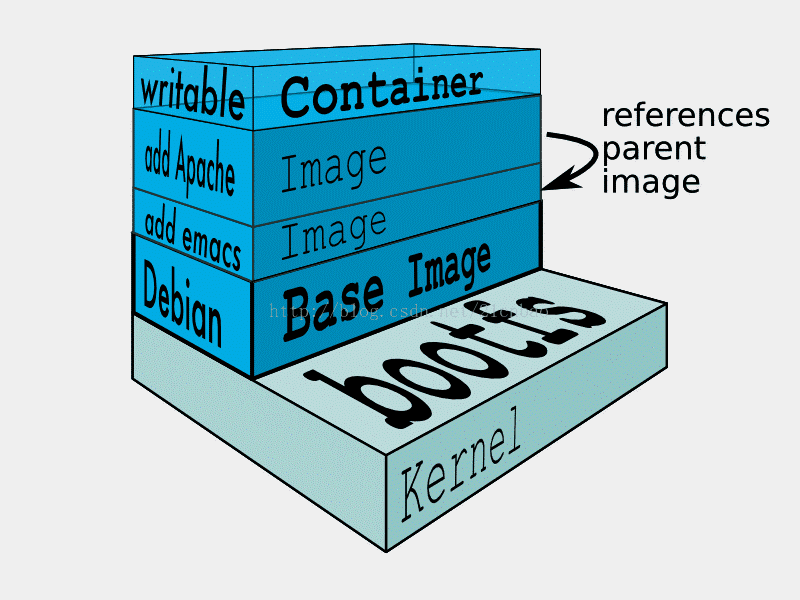


一般的流程中,client发pull命令从仓库把image拉到docker host，然后通过run命令指挥image到host上面弄一个container来跑这个image。

当然也可以是相反的流程，client 通过build命令在host上面创建一个自己的image，然后通过push命令把image推到仓库。之后这个image可以被别的人或者自己pull。

image到底是个什么鬼？

Docker镜像是一个特殊的文件系统，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）。image为特定目的而生，比如弄了个nginx的image后，这个image就把nginx的东西包罗万象了，无论是张三、王五、六麻子还是七癞子，无论它是什么电脑，什么操作系统，只要支持docker，它把这个nginx的image下载下来后，拿docker run命令就可以弄容器跑nginx了。这样，用户就不用装nginx以及它依赖的一切包了（通常装一个软件弄依赖也能把你弄地烦躁死了）。这样看起来，Docker实在是居家旅行，杀人越货之必备良器也！



镜像构建时，会一层层叠加，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为该文件已删除。所以这个很类似[**Git**](http://lib.csdn.net/base/28)里面这一次提交相对于上一次提交的diff：

1. https

关于http的发送

blog: http://blog.csdn.net/hudashi/article/details/50789006

http请求由三部分组成：请求行、 请求头和请求正文。

请求行就是第一行，是“方法 URL 协议/版本”，并以 回车换行作为结尾。请求行以空格分隔。格式如下：

POST /index.php HTTP/1.1

以上代码中“POST”代表请求方法，“/index.php”表示URI，“HTTP/1.1代表协议和协议的版本

还有其他的方法，但是常用的有POST和GET。

GET 方法用于获取由 Request-URI 所标识的资源的信息，GET方法是默认的HTTP请求方法，例如当我们通过在浏览器的地址栏中直接输入网址的方式去访问网页的时候，浏览器采用的就是 GET 方法向服务器获取资源。我们可以使用GET方法来提交表单数据，用GET方法提交的表单数据只经过了简单的编码，同时它将作为URL的一部分向服务器发送，因此，如果使用GET方法来提交表单数据就存在着安全隐患上。例如：

Http://localhost/login.php?username=aa&password=1234

从上面的URL请求中，很容易就可以辩认出表单提交的内容。（？之后的内容）另外由于GET方法提交的数据是作为URL请求的一部分所以提交的数据量不能太大。这是因为浏览器对url的长度有限制

各种浏览器也会对url的长度有所限制，下面是几种常见浏览器的url长度限制:(单位:字符)

IE : 2803

Firefox:65536

Chrome:8182

Safari:80000

Opera:190000

POST的方法是对于Get方法的替代方法。它主要是向Web服务器提交表单数据，尤其是大批量的数据。 在请求头信息结束之后的两个回车换行之后（实际是空一行），就是表单提交的数据

POST方法克服了GET方法的一些缺点。通过POST方法提交表单数据时，数据不是作为URL请求的一部分而是作为标准数据传送给Web服务器，这就克服了GET方法中的信息无法保密和数据量太小的缺点。因此，出于安全的考虑以及对用户隐私的尊重，通常表单提交时采用POST方法。

　 从编程的角度来讲，如果用户通过GET方法提交数据，则数据存放在QUERY＿STRING环境变量中，而POST方法提交的数据则可以从标准输入流中获取。

GET与POST方法有以下区别：

1、 在客户端，Get方式在通过URL提交数据，数据在URL中可以看到；POST方式，数据放在HTTP包的body中。

2、 GET方式提交的数据大小有限制（因为浏览器对URL的长度有限制），而POST则没有此限制。

3、安全性问题。正如在（1）中提到，使用 Get 的时候，参数会显示在地址栏上，而 Post 不会。所以，如果这些数据是中文数据而且是非敏感数据，那么使用 get；如果用户输入的数据不是中文字符而且包含敏感数据，那么还是使用 post为好。

4.、服务器取值方式不一样。GET方式取值，如php可以使用$\_GET来取得变量的值，而POST方式通过$\_POST来获取变量的值。

然后是请求头，每个头域由一个域名，冒号（:）和域值三部分组成。域名是大小写无关的，域值前可以添加任何数量的空格符，头域可以被扩展为多行，在每行开始处，使用至少一个空格或制表符。

头域包括Transport头域，Client头域，Cookie头域，Entity头域等等。

在接收和解释请求消息后，服务器会返回一个 HTTP 响应消息。与 HTTP 请求类似，HTTP 响应也是由三个部分组成，分别是：状态行、消息报头和响应正文。

状态行由协议版本、数字形式的状态代码，及相应的状态描述组成，各元素之间以空格分隔，结尾时回车换行符，

状态代码与状态描述

状态代码由 3 位数字组成， 表示请求是否被理解或被满足，状态描述给出了关于状态码的简短的文字描述。状态码的第一个数字定义了响应类别，后面两位数字没有具体分类。第一个数字有 5 种取值，如下所示。

1xx：指示信息——表示请求已经接受，继续处理

2xx：成功——表示请求已经被成功接收、理解、接受。

3xx：重定向——要完成请求必须进行更进一步的操作

4xx：客户端错误——请求有语法错误或请求无法实现

5xx：服务器端错误——服务器未能实现合法的请求。

常见状态代码、状态描述、说明：

200 OK //客户端请求成功

400 Bad Request //客户端请求有语法错误，不能被服务器所理解

401 Unauthorized //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用

403 Forbidden //服务器收到请求，但是拒绝提供服务

404 Not Found //请求资源不存在，eg：输入了错误的URL

500 Internal Server Error //服务器发生不可预期的错误

503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

1. threading

python中的threading包

blog: http://blog.csdn.net/eastmount/article/details/50155353