

**Proxy**: Harbor的registry、UI、token services等组件，都处在一个反向代理后边。该代理将来自浏览器、docker clients的请求转发到后端服务上

**Registry**: 负责存储Docker镜像，以及处理Docker push/pull请求。因为Harbor强制要求对镜像的访问做权限控制， 在每一次push/pull请求时，Registry会强制要求客户端从token service那里获得一个有效的token

**Core services**: Harbor的核心功能，主要包括如下3个服务:

**UI**: 作为Registry Webhook, 以图像用户界面的方式辅助用户管理镜像。1) WebHook是在registry中配置的一种机制， 当registry中镜像发生改变时，就可以通知到Harbor的webhook endpoint。Harbor使用webhook来更新日志、初始化同步job等。 2) Token service会根据该用户在一个工程中的角色，为每一次的push/pull请求分配对应的token。假如相应的请求并没有包含token的话，registry会将该请求重定向到token service。 3) Database 用于存放工程元数据、用户数据、角色数据、同步策略以及镜像元数据。

**Job services**: 主要用于镜像复制，本地镜像可以被同步到远程Harbor实例上。

**Log collector**: 负责收集其他模块的日志到一个地方

这里我们与上面运行的7个容器对比，对harbor-adminserver感觉有些疑虑。其实这里harbor-adminserver主要是作为一个后端的配置数据管理，并没有太多的其他功能。harbor-ui所要操作的所有数据都通过harbor-adminserver这样一个数据配置管理中心来完成。

**Harbor实现**

Harbor的每一个组件都被包装成一个docker容器。自然，Harbor是通过docker compose来部署的。在[Harbor源代码](https://github.com/vmware/harbor)的make目录下的docker-compose模板会被用于部署Harbor。打开该模板文件，可以看到Harbor由7个容器组件所组成

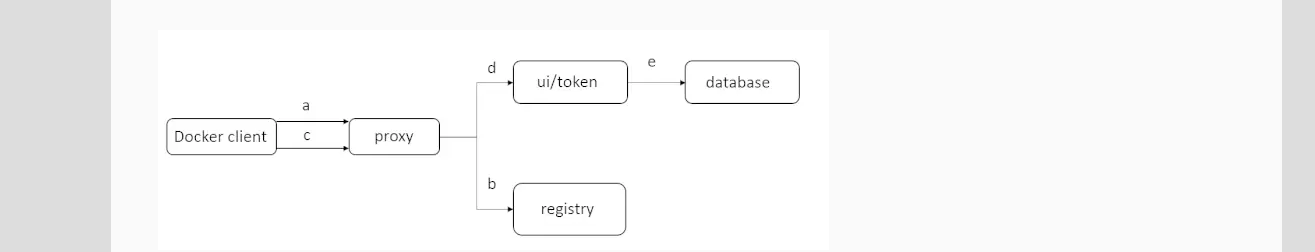
* proxy: 通过nginx服务器来做反向代理
* registry: docker官方发布的一个仓库镜像组件
* ui: 整个架构的核心服务。该容器是Harbor工程的主要部分
* adminserver: 作为Harbor工程的配置数据管理器使用
* mysql: 通过官方Mysql镜像创建的数据库容器
* job services: 通过状态机的形式将镜像复制到远程Harbor实例。镜像删除同样也可以被同步到远程Harbor实例中。
* log: 运行rsyslogd的容器，主要用于收集其他容器的日志
* 这些容器之间都通过Docker内的DNS服务发现来连接通信。通过这种方式，每一个容器都可以通过相应的容器来进行访问。对于终端用户来说，只有反向代理(Nginx)服务的端口需要对外暴露。
* 如下两个Docker命令行的例子显示了Harbor各组件之间的交互

## docker login处理流程

假设Harbor被部署在192.168.1.10宿主机上。用户通过如下命令行发送一个登录Harbor的请求：

docker login 192.168.1.10

在用户输入所需要的Credentials之后，Docker Client会发送一个HTTP GET请求到192.168.1.10/v2/地址处，Harbor的不同容器组件将会按照如下步骤进行处理：



1. 首先，该请求将会被监听在80端口上的代理容器所接收到。容器中的Nginx将会把该请求转发给后端的Registry容器

b) 由于registry容器已经被配置为基于token的认证，因此其会返回一个401错误码，用于通知docker客户端从一个指定的URL处获得一个有效的token。在Harbor中，该URL会指向Core service中的token service。

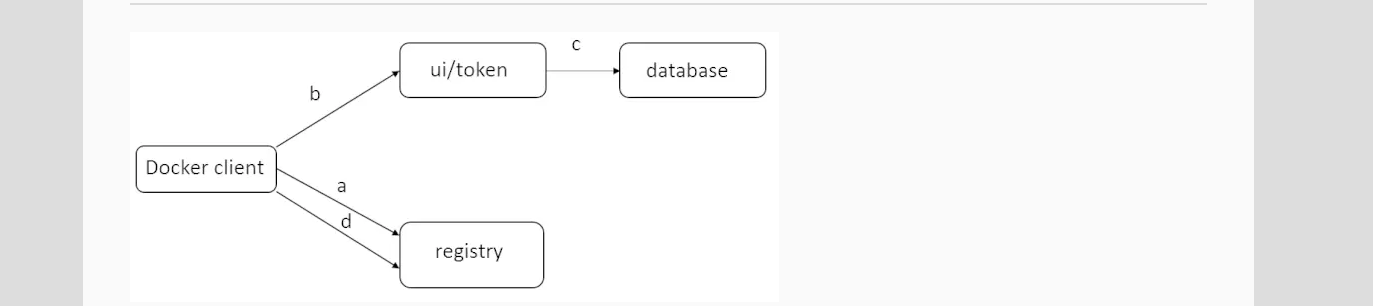
(c) 当Docker Client接收到这个错误码，其就会发送一个请求到token service URL，会根据HTTP基本认证协议在请求头中内嵌username和password相关信息

(d) 在该请求被发送到代理的80端口上后，Nginx会根据预先所配置的规则将请求转发到UI容器上。UI容器中的token service接收到该请求之后，其就会对该请求进行解码然后获得相应的用户名及密码

(e) 在成功获得用户名及密码之后，token Service就会检查mysql数据库以完成用户的认证。当token service被配置为LDAP/AD认证的时候，其就会通过外部的LDAP/AD服务来完成认证。在成功认证之后，token Service就会返回一个认证成功的http code， Http body部分会返回一个通过private key所产生的token

到这里为止，Docker login就处理完成。Docker client会将步骤(c)所产生的username及password编码后保存到一个隐藏的文件中

## docker push处理流程



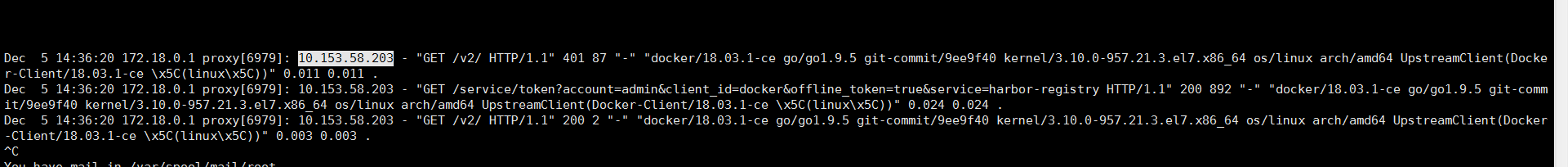
注： 上述我们已经省略了Nginx代理转发的步骤。上图展示了docker push过程中不同组件之间的交互流程）

在用户成功登录之后，就会通过Docker push命令向Harbor发送一个Docker Image:

# docker push 192.168.1.10/library/hello-world

1. 首先，docker client执行类似登录时的流程发送一个请求到registry，然后返回一个token service的URL
2. 然后，docker client通过提供一些额外的信息与ui/token交互以获得push镜像library/hello-world的token
3. 在成功获得来自Nginx转发的请求之后，Token Service查询数据库以寻找用户推送镜像的角色及权限。假如用户有相应的权限，token service就会编码相应的push操作信息，并用一个private key进行签名。然后返回一个token给Docker client
4. 在docker client获得token之后，其就会发送一个push请求到registry，在该push请求的头中包含有上面返回的token信息。一旦registry收到了该请求，其就会使用public key来解码该token，然后再校验其内容。该public key对应于token service处的private key。假如registry发现该token有效，则会开启镜像的传输流程

<https://youendless.com/post/docker_login_process/>



<http://106.74.18.236:34976/service/token?account=admin&client_id=docker&offline_token=true&service=harbor-registry>

GET http://106.74.18.236:34976/service/token?account=admin&client\_id=docker&offline\_token=true&service=harbor-registry

Authorization: Basic YWRtaW46SHRLZyVeQmpodHp6LktKeTEyMw==

<https://youendless.com/post/docker_login_process/>

curl https://harbor.casicloud.com/v2 -k -IL

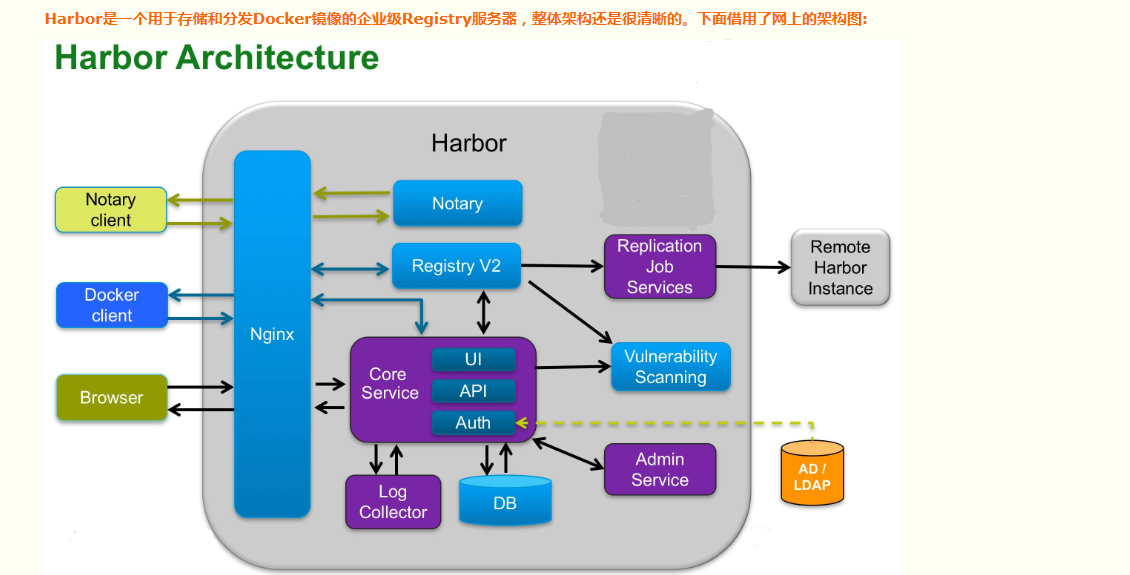
docker 认证源码分析

<http://dockone.io/article/845>

docker认证原理

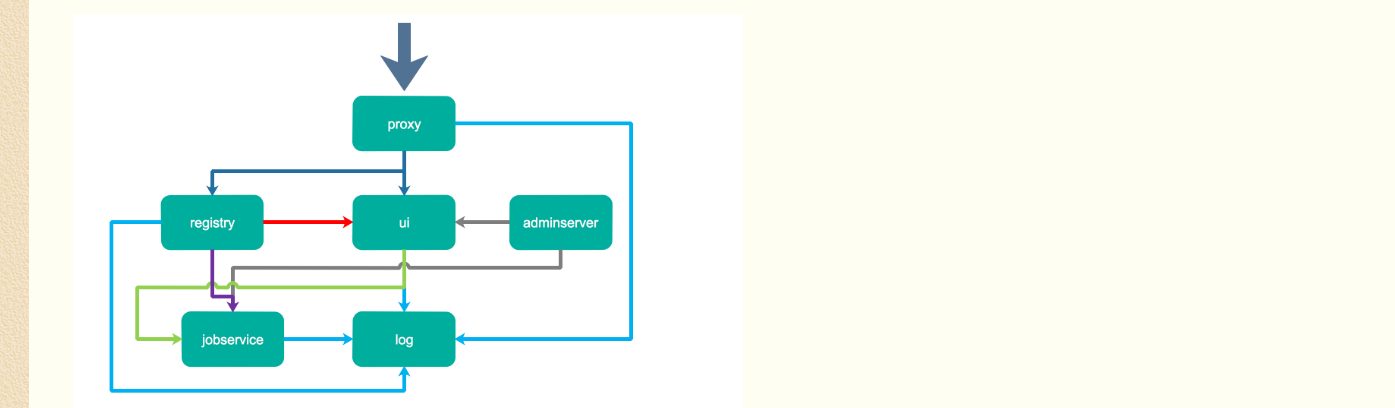
<https://mojo-zd.github.io/2018/04/12/Docker-Registry-V2%E8%AE%A4%E8%AF%81%E6%9C%BA%E5%88%B6%EF%BC%8C%E4%BB%A5%E5%8F%8A%E5%A6%82%E4%BD%95%E6%9E%84%E5%BB%BA%E8%87%AA%E5%B7%B1%E7%9A%84Registy-Auth-Server/>

Harbor的每个组件都是以Docker容器的形式构建的，可以使用Docker Compose来进行部署。如果环境中使用了kubernetes，Harbor也提供了kubernetes的配置文件。**Harbor大概需要以下几个容器组成**：**ui**(Harbor的核心服务)、**log**(运行着rsyslog的容器，进行日志收集)、**mysql**(由官方mysql镜像构成的数据库容器)、**Nginx**(使用Nginx做反向代理)、**registry**(官方的Docker registry)、**adminserver**(Harbor的配置数据管理器)、**jobservice**(Harbor的任务管理服务)、**redis**(用于存储session)。



Nginx(即Proxy代理层): Nginx前端代理，主要用于分发前端页面ui访问和镜像上传和下载流量; Harbor的registry,UI,token等服务，通过一个前置的反向代理统一接收浏览器、Docker客户端的请求，并将请求转发给后端不同的服务。  
**->** Registry v2: 镜像仓库，负责存储镜像文件; Docker官方镜像仓库, 负责储存Docker镜像，并处理docker push/pull命令。由于我们要对用户进行访问控制，即不同用户对Docker image有不同的读写权限，Registry会指向一个token服务，强制用户的每次docker pull/push请求都要携带一个合法的token, Registry会通过公钥对token进行解密验证。  
**->** Database(MySQL或Postgresql)：为core services提供数据库服务，负责储存用户权限、审计日志、Docker image分组信息等数据。

**Harbor自有组件**  
**->** Core services(Admin Server): 这是Harbor的核心功能，主要提供以下服务：   
    **->** UI：提供图形化界面，帮助用户管理registry上的镜像（image）, 并对用户进行授权。  
    **->** webhook：为了及时获取registry 上image状态变化的情况， 在Registry上配置webhook，把状态变化传递给UI模块。  
    **->** Auth服务：负责根据用户权限给每个docker push/pull命令签发token. Docker 客户端向Regiøstry服务发起的请求,如果不包含token，会被重定向到这里，获得token后再重新向Registry进行请求。  
   **->** API: 提供Harbor RESTful API  
**->** Replication Job Service：提供多个 Harbor 实例之间的镜像同步功能。  
**->** Log collector：为了帮助监控Harbor运行，负责收集其他组件的log，供日后进行分析。



**->** proxy，它是一个nginx前端代理，主要是分发前端页面ui访问和镜像上传和下载流量，上图中通过深蓝色先标识；  
**->** ui提供了一个web管理页面，当然还包括了一个前端页面和后端API，底层使用mysql数据库；  
**->** registry是镜像仓库，负责存储镜像文件，当镜像上传完毕后通过hook通知ui创建repository，上图通过红色线标识，当然registry的token认证也是通过ui组件完成；  
**->** adminserver是系统的配置管理中心附带检查存储用量，ui和jobserver启动时候回需要加载adminserver的配置，通过灰色线标识；  
**->** jobsevice是负责镜像复制工作的，他和registry通信，从一个registry pull镜像然后push到另一个registry，并记录job\_log，上图通过紫色线标识；  
**->** log是日志汇总组件，通过docker的log-driver把日志汇总到一起，通过浅蓝色线条标识。

**->** log是日志汇总组件，通过docker的log-driver把日志汇总到一起，通过浅蓝色线条标识。

**Harbor的误区**  
**误区一:** Harbor是负责存储容器镜像的 (Harbor是镜像仓库，那么它就应当是存储镜像的)  
其实关于镜像的存储，Harbor使用的是官方的docker registry服务去完成，至于registry是用本地存储或者s3都是可以的，Harbor的功能是在此之上提供用户权限管理、镜像复制等功能，提高使用的registry的效率。

**误区二**：Harbor镜像复制是存储直接复制 (镜像的复制，很多人以为应该是镜像分层文件的直接拷贝)  
其实Harbor镜像复制采用了一个更加通用、高屋建瓴的做法，通过docker registry 的API去拷贝，这不是省事，这种做法屏蔽了繁琐的底层文件操作、不仅可以利用现有docker registry功能不必重复造轮子，而且可以解决冲突和一致性的问题。

**Harbor的部署**  
这里不建议使用kubernetes来部署, 原因是镜像仓库非常重要, 尽量保证部署和维护的简洁性, 因此这里直接使用compose的方式进行部署。官方提供3种部署Harbor的方式:  
1）**在线安装**: 从Docker Hub下载Harbor的镜像来安装, 由于Docker Hub比较慢, 建议Docker配置好加速器。  
2）**离线安装**: 这种方式应对与部署主机没联网的情况使用。需要提前下载离线安装包: harbor-offline-installer-.tgz 到本地  
3）**OVA安装**: 这个主要用vCentor环境是使用

后面部署时会为Docker配置镜像加速器, 因此会采用在线部署的方式, 部署步骤如下：  
-> 下载Harbor最新的在线安装包  
-> 配置Harbor (harbor.cfg)  
-> 运行install.sh来安装和启动Harbor  
-> Harbor的日志路径:/var/log/harbor

Harbor仓库部署的官方要求的最小系统配置  
-> 2个cpu  
-> 4g内存  
-> 40g硬盘，因为是存储镜像的所以推荐硬盘大点。

Habor 配置https问题

<https://blog.csdn.net/watermelonbig/article/details/82667497>