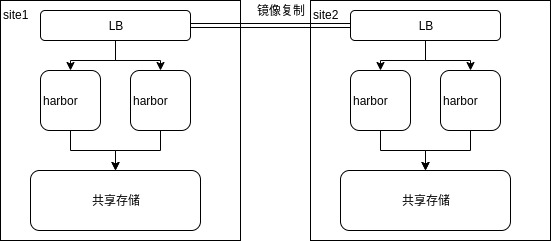
# Harbor

## 部署图

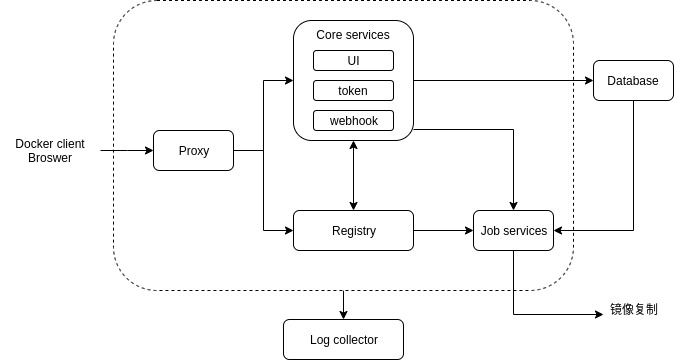


## 组件图

Harbor在架构上主要由6个组件构成：

1. Proxy：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。
2. Registry： 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。
3. Core services： 这是Harbor的核心功能，主要提供以下服务：
4. 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。
5. 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。
6. token 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。
7. Database：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。我们采用外部数据库。
8. Job Services：提供镜像远程复制功能，可以把本地镜像同步到其他Harbor实例中。
9. 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。

各个组件之间的关系如下图所示：



## 容量规划

生产环境推荐4C16G的服务器来部署一个habor实例。共享存储推荐500G以上。

存储大小和镜像的多少，镜像大小以及保留版本有关。参考：<https://docs.docker.com/ee/dtr/admin/install/system-requirements/>

注: 以下是在4C8G的个人电脑上的测试结果。client和harbor部署在统一台机器上。

并发数为1，10，100的pull耗时。

并发数为1，10，100的push耗时。

## 部署

1. 提前挂在共享储存到待部署服务器的/data目录。
2. 创建外置的PostgreSQL数据库。
3. 部署harbor

# git clone <https://gitlab.oneitfarm.com/hl/harbor.git>

# cd harbor

# vim custom.cfg

根据实际环境信息修改配置文件。

# domain name

\_\_DOMAIN=test.harbor.com

\_\_UI\_URL\_PROTOCOL=https

\_\_WORK\_DIR=/data/

\_\_CERT\_DIR=cert/

# admin password

\_\_HARBOR\_ADMIN\_PASSWORD=Harbor12345

#The address of the Harbor database. Only need to change when using external db.

\_\_DB\_HOST=192.168.3.2

#The password for the root user of Harbor DB. Change this before any production use.

\_\_DB\_PASSWORD=123456

#The port of Harbor database host

\_\_DB\_PORT=5432

#The user name of Harbor database

\_\_DB\_USER=postgres

# ./prepare.sh

# ./install.sh

## 压力测试

# git clone <https://gitlab.oneitfarm.com/hl/harbor.git>

# cd harbor

修改test.py里的并发数，habor信息

# python test.py

在测试目录下会生成测试报告。后缀为csv的是测试结果。