Docker Harbor 调研文档

作者:洪必梁

时间:2019.1.2

调研背景

当前需求

由于当前部署了多个的docker registry仓库，仓库间的同步操作非常麻烦。如果要同步仓库A的镜像到仓库B上，则需要手动在仓库B上pull相应的镜像，之后再打上tag，push到仓库B,操作繁琐,浪费时间精力。 因此，为了解决该问题，就想寻求能够直接将仓库A的镜像推送到仓库B的解决方案。

经过一轮的调研之后发现docker registry并没有什么api或者命令能够直接做到上面所描述的操作。故而就放宽搜索限制，不再局限于docker registry上的解决方案。在这过程中发现了Harbor这个企业级docker仓库解决方案，发现其特性能够满足需求，并且能够做的更好。**Harbor可以以项目为单位，将项目里面所包含的所有镜像，由项目A同步到项目B。**故而对Harbor展开更深入的调研

Harbor简介

Harbor是一个用于存储和分发Docker镜像的企业级Registry服务器，通过添加一些企业必需的功能特性，例如安全、标识和管理等，扩展了开源Docker Distribution。作为一个企业级私有Registry服务器，Harbor提供了更好的性能和安全。提升用户使用Registry构建和运行环境传输镜像的效率。Harbor支持安装在多个Registry节点的镜像资源复制，镜像全部保存在私有Registry中， 确保数据和知识产权在公司内部网络中管控。另外，Harbor也提供了高级的安全特性，诸如用户管理，访问控制和活动审计等。

特性

> （1）基于角色的访问控制 ： 用户与Docker镜像仓库通过“项目”进行组织管理，一个用户可以对多个镜像仓库在同一命名空间（project）里有不同的权限。

> （2）镜像复制 ： 镜像可以在多个Registry实例中复制（同步）。尤其适合于负载均衡，高可用，混合云和多云的场景。

> （3）图形化用户界面 ： 用户可以通过浏览器来浏览，检索当前Docker镜像仓库，管理项目和命名空间。

> （4）AD/LDAP 支持 ： Harbor可以集成企业内部已有的AD/LDAP，用于鉴权认证管理。

> （5）审计管理 ： 所有针对镜像仓库的操作都可以被记录追溯，用于审计管理。

> （6）国际化 ： 已拥有英文、中文、德文、日文和俄文的本地化版本。更多的语言将会添加进来。

> （7）RESTful API ： RESTful API 提供给管理员对于Harbor更多的操控, 使得与其它管理软件集成变得更容易。

> （8）部署简单 ： 提供在线和离线两种安装工具， 也可以安装到vSphere平台(OVA方式)虚拟设备。

组件

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。

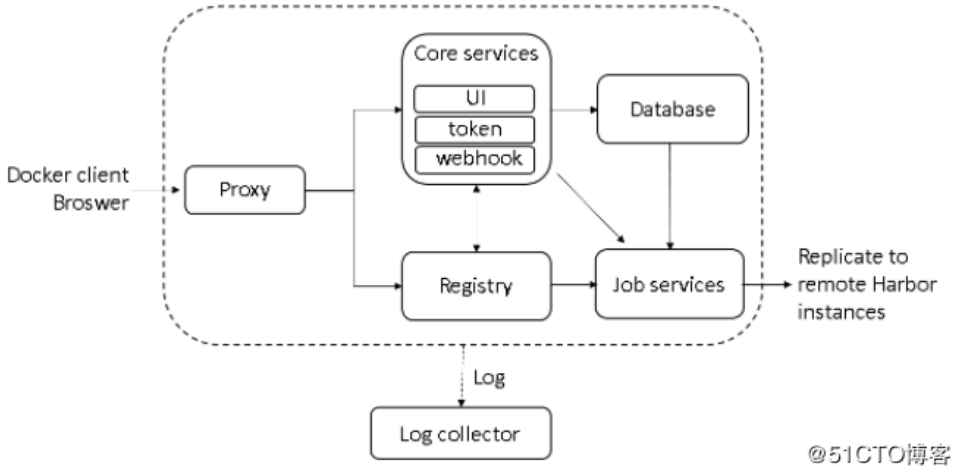
> （3）Core services： 这是Harbor的核心功能，主要提供以下服务： 1）UI：提供图形化界面，帮助用户管理registry上的镜像（image），并对用户进行授权。 2）webhook：为了及时获取registry 上image状态变化的情况， 在Registry上配置webhook，把状态变化传递给UI模块。 3）token 服务：负责根据用户权限给每个docker push/pull命令签发token. Docker客户端向Regiøstry服务发起的请求,如果不包含token，会被重定向到这里，获得token后再重新向Registry进行请求。

> （4）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。

> （5）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。

> （6）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：



Harbor构建

Harbor的每个组件都是以Docker容器的形式构建的，官方也是使用Docker Compose来对它进行部署。用于部署Harbor的Docker Compose模板位于 harbor/docker-compose.yml,打开这个模板文件，发现Harbor是由7个容器组成的；

（1）nginx：nginx负责流量转发和安全验证，对外提供的流量都是从nginx中转，所以开放https的443端口，它将流量分发到后端的ui和正在docker镜像存储的docker registry。

（2）harbor-jobservice：harbor-jobservice 是harbor的job管理模块，job在harbor里面主要是为了镜像仓库之前同步使用的;

（3）harbor-ui：harbor-ui是web管理页面，主要是前端的页面和后端CURD的接口;

（4）registry：registry就是docker原生的仓库，负责保存镜像。

（5）harbor-adminserver：harbor-adminserver是harbor系统管理接口，可以修改系统配置以及获取系统信息。

（6）harbor-db：harbor-db是harbor的数据库，这里保存了系统的job以及项目、人员权限管理。由于本harbor的认证也是通过数据，在生产环节大多对接到企业的ldap中；

（7）harbor-log：harbor-log是harbor的日志服务，统一管理harbor的日志。通过inspect可以看出容器统一将日志输出的syslog。

这几个容器通过Docker link的形式连接在一起，这样，在容器之间可以通过容器名字互相访问。**对终端用户而言，只需要暴露proxy （即Nginx）的服务端口。**

说完了原理性的东西，现在我们动手实际操作一下，亲自感受harbor带来的变化。

Docker Harbor 安装

> 注：该文档的工作环境为centos7 docker版本17.03

目标主机需要安装Python，Docker和Docker Compose 其中Python有centos7本身自带，不需要进行额外安装

安装docker-compose

```
yum install docker-compose -y
```

接下来就说harbor的安装，Harbor安装步骤归结为以下几点：

> 下载安装程序;

> 配置harbor.cfg文件;

> 运行install.sh来安装并启动Harbor;

下载地址：

```
https://github.com/goharbor/harbor/releases
```

当前文档使用的是v1.7.0 找到对应版本，选择Harbor offline installer进行下载

这里选择的是离线版本，安装的命令如下：

```
tar xvf harbor-offline-installer-v1.7.0.tgz  -C /opt/
```

接下来进入到/opt/harbor/进行配置工作

配置参数位于文件harbor.cfg中。

harbor.cfg中有两类参数，所需参数和可选参数。

> 所需参数：这些参数需要在配置文件harbor.cfg中设置，如果用户更新它们并运行install.sh脚本重新安装Harbour，参数将生效。

> 可选参数：这些参数对于更新是可选的，即用户可以将其保留为默认值，并在启动Harbour后在Web UI上进行更新。如果他们进入harbor.cfg，他们只会在第一次启动Harbor时生效，随后对这些参数的更新，harbor.cfg将被忽略。

所需参数：

> hostname：用于访问用户界面和register服务。它应该是目标机器的IP地址或完全限定的域名（FQDN），例如192.168.1.10或reg.yourdomain.com。不要使用localhost或127.0.0.1为主机名。

> ui_url_protocol：（http或https，默认为http）用于访问UI和令牌/通知服务的协议。如果公证处于启用状态，则此参数必须为https。

> max_job_workers：镜像复制作业线程。

> db_password：用于db_auth的MySQL数据库的root密码。

> customize_crt：打开或关闭，默认打开）打开此属性时，准备脚本创建私钥和根证书，用于生成/验证注册表令牌。当由外部来源提供密钥和根证书时，将此属性设置为off。

> ssl_cert：SSL证书的路径，仅当协议设置为https时才应用。

> ssl_cert_key：SSL密钥的路径，仅当协议设置为https时才应用。

> secretkey_path：用于在复制策略中加密或解密远程register密码的密钥路径。

可选参数：

> 电子邮件设置：Harbor需要这些参数才能向用户发送"密码重置"电子邮件，并且只有在需要该功能时才需要。另外，请注意，在默认情况下SSL连接时没有启用-如果你的SMTP服务器需要SSL，但不支持STARTTLS，那么你应该通过设置启用SSL email_ssl = TRUE。

> harbour_admin_password：管理员的初始密码，这个密码只在Harbour第一次启动时生效。之后，此设置将被忽略，并且应在UI中设置管理员的密码。请注意，默认的用户名/密码是admin/Harbor12345。

> auth_mode：使用的认证类型，默认情况下，它是db_auth，即凭据存储在数据库中。对于LDAP身份验证，请将其设置为ldap_auth。

> self_registration：（打开或关闭，默认打开）启用/禁用用户注册功能。禁用时，新用户只能由Admin用户创建，只有管理员用户可以在Harbour中创建新用户。 注意：当auth_mode设置为ldap_auth时，自注册功能将始终处于禁用状态，并且该标志被忽略。

> token_expiration：由令牌服务创建的令牌的到期时间（分钟），默认为30分钟。 project_creation_restriction：用于控制哪些用户有权创建项目的标志。默认情况下，每个人都可以创建一个项目，设置为“adminonly”，这样只有admin可以创建项目。

> verify_remote_cert：（打开或关闭，默认打开）此标志决定了当Harbour与远程register实例通信时是否验证SSL/TLS证书。将此属性设置为off将绕过SSL/TLS验证，这在远程实例具有自签名或不可信证书时经常使用。

由于启动harbor之后会开启很多容器，并且包含如下容器:

```
        NAMES
3/tcp   nginx
        harbor-jobservice
        harbor-portal
        harbor-core
        harbor-adminserver
        harbor-db
        registry
        registryctl
        redis
        harbor-log
```

本地不能有名称相同的容器，可以使用命令：

```
docker rename oldName newName
```

方式进行修改，也可以修改/opt/harbor/docker-compose.yml进行容器名称的修改。

如果要修改容器暴露的端口号等信息，则需要对docker-compose.yml进行更加详细的配置修改了。

配置完成就可以启动Harbor了，如下操作：

```
./install.sh
```

提示完成之后，可以使用docker-compose命令进行查看(需在/opt/harbor目录下)：

```
docker-compose ps
```

可以看到如下内容：

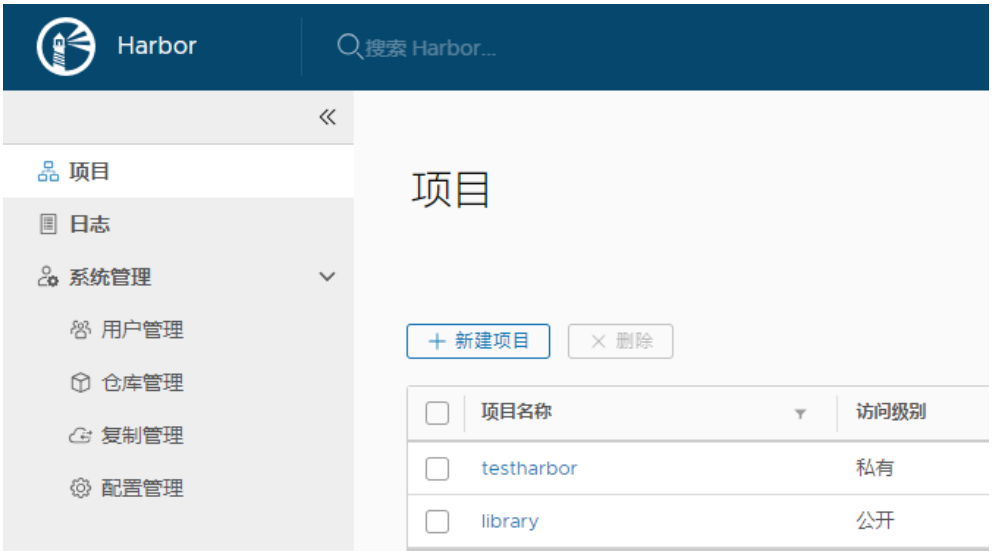
```
[root@localhost harbor]# docker-compose ps
      Name                    Command               State                                 Ports
-----------------------------------------------------------------------------------------------------------------------------------
harbor-adminserver   /harbor/start.sh                 Up
harbor-core          /harbor/start.sh                 Up
harbor-db            /entrypoint.sh postgres          Up      5432/tcp
harbor-jobservice    /harbor/start.sh                 Up
harbor-log           /bin/sh -c /usr/local/bin/ ...   Up      127.0.0.1:1514->10514/tcp
harbor-portal        nginx -g daemon off;             Up      80/tcp
nginx                nginx -g daemon off;             Up      0.0.0.0:443->443/tcp, 0.0.0.0:4443->4443/tcp, 0.0.0.0:80->80/tcp
redis                docker-entrypoint.sh redis ...   Up      6379/tcp
registry             /entrypoint.sh /etc/regist ...   Up      5000/tcp
registryctl          /harbor/start.sh                 Up
```

如果到目前为止一切顺利的话，就可以进行web访问了。输入上面配置的hostname到浏览器，使用admin/Harbor12345进行登陆。

登陆完成之后要马上修改admin的密码。

接下来创建一个project来验证我们的安装是否成功了。

我这边创建了一个叫做testharbor的项目



访问级别请选择成私有的，不然别人就能不需要登陆就可以进行访问了。

由于Harbor的默认安装使用HTTP，而Register v2版本开始必须使用HTTPS，因此你需要将该选项添加 --insecure-registry到客户端的Docker守护程序并重新启动Docker服务。

所以在你的测试的客户机上面要进行相应的修改才能正常推拉镜像

```
vi /lib/systemd/system/docker.service
#在ExecStart值 在末尾添加，hostname为前面配置的值
--insecure-registry=hostname

//重启docker
systemctl daemon-reload
systemctl restart docker
```

此时可以进行进行推拉测试了。 首先，将本地的一个镜像加上tag，如：

```
docker tag mysql hostname/testharbor/mysql:latest
```

其中 hostname是之前配置的服务器地址，testharbor是刚刚创建的项目名称

在push给服务器之前，我们需要做一下登陆操作

```
docker login -u admin -p Harbor12345 http://hostname
```

如果没做上面那步修改docker配置文件的insecure-registry话，登陆的时候会报错

登陆成功之后就可以进行push了

```
docker push hostname/testharbor/mysql:latest
```

如果这步也成功，说明目前已经完全安装成功了。

多仓库镜像同步

这边的同步指的是项目级别的同步，将特定的项目同步到另外一个harbor仓库上面去。

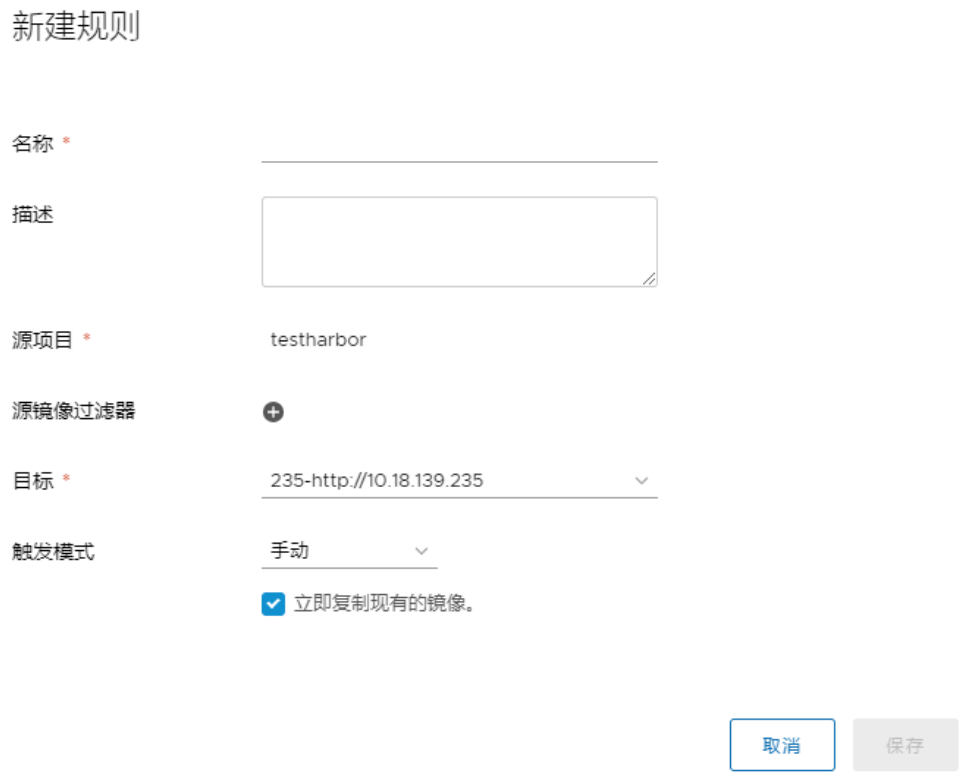
按照上面的步骤，重新在另外一台主机上安装好harbor仓库。上面那台harbor仓库假定为harbor A ，后面创建的这边为harbor B。

登入到harbor A后台，选择之前创建的那个testharbor工程

选择复制tab，并点击下面的创建规则。 创建规则页面下有个目标的选项，点击之后进入到仓库管理页面，新增一个目标。

填上相应的内容，目标url填写http://hostnameB 用户名密码也填好，这边验证远程证书就去掉勾选

回到创建规则页面，开始创建规则：



这里的触发模式的话有3种，分别为即刻，手动，定时根据自己的需要进行选择。 填上相应的内容之后，点击下面的测试连接，如果成功再点击确定。

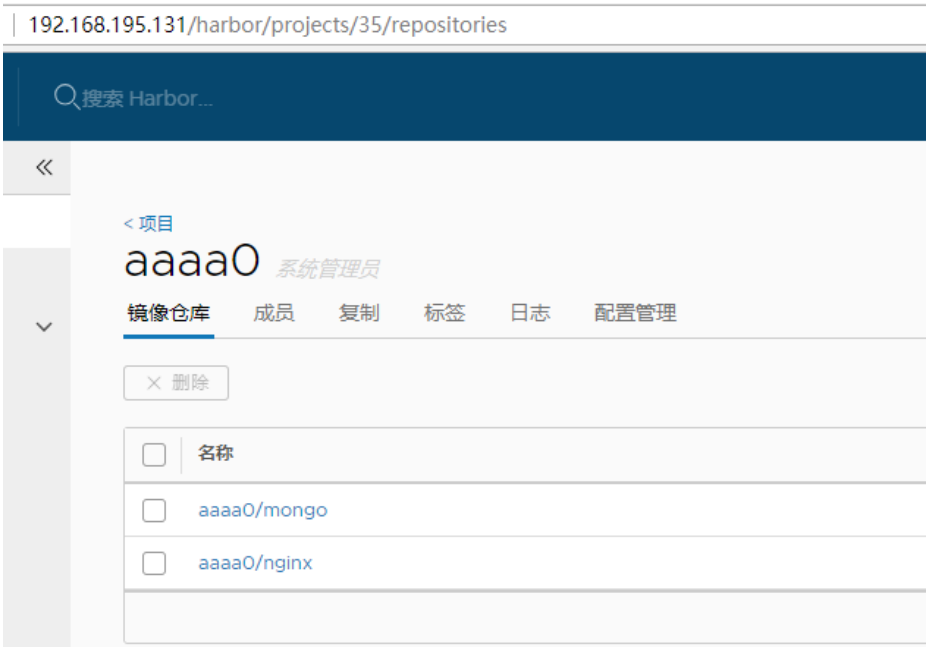
由于我这边创建的时候选择的是即刻，创建完毕之后，马上登陆到harbor B的后台进行查看， 如果看到testharbor项目以及里面的镜像同步过来了，则说明镜像同步的功能已经完成了。

完整操作流程

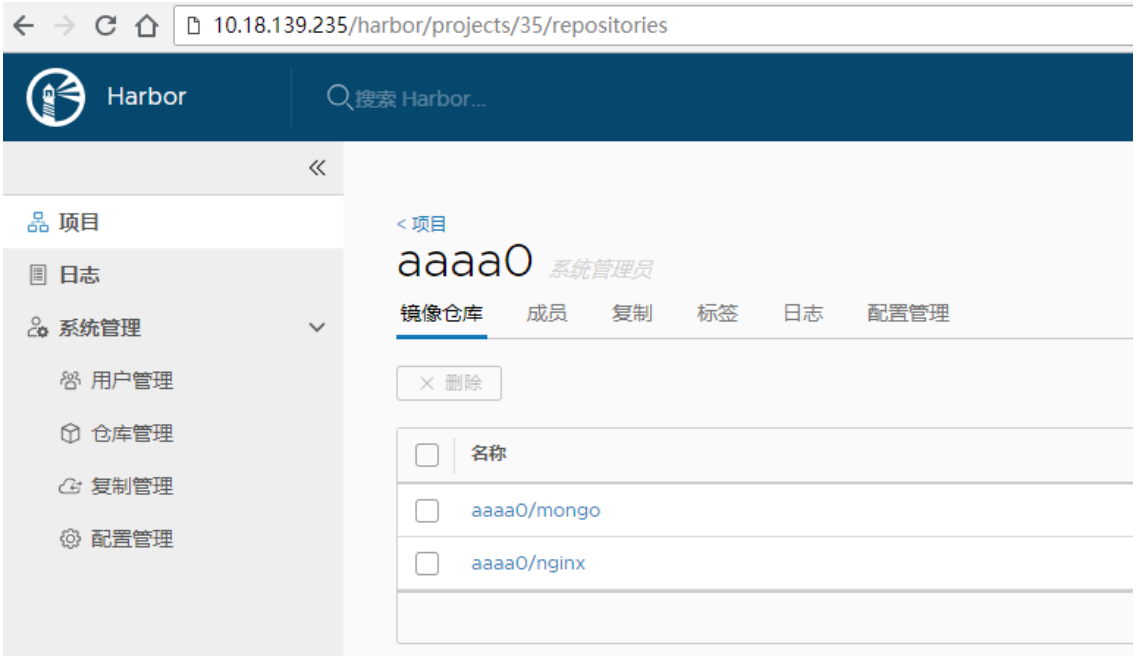
由客户端登陆到仓库A，并将本地镜像重命名，再推送到仓库A的aaaa0项目下

```
[root@allen ~]# docker login -u admin -p Starnet0591 http://192.168.195.131
Login Succeeded
[root@allen ~]#
[root@allen ~]#
[root@allen ~]# docker tag nginx 192.168.195.131/aaaa0/nginx:latest
[root@allen ~]# docker push 192.168.195.131/aaaa0/nginx:latest
The push refers to a repository [192.168.195.131/aaaa0/nginx]
77e23640b533: Mounted from testharbor2/nginx
757d7bb101da: Mounted from testharbor2/nginx
3358360aedad: Mounted from testharbor2/nginx
latest: digest: sha256:a08ed346dfbb55cf7819dbe60f574f19fe387f2e7486cdc2073f1272d1344ec9 size: 948
```

推送完成之后，再到仓库A的web界面进行查看， 可以看到nginx这个镜像已经推送到仓库A的aaaa0项目下了。



再使用上面的复制操作，登陆到仓库B的后台服务器上来。 可以看到，项目跟镜像全部都同步过来了。



管理命令

你可以使用docker-compose来管理Harbor。一些有用的命令如下所示（必须在与docker-compose.yml相同的目录中运行）。

停止/启动/重启Harbor：

```
docker-compose stop
docker-compose start
docker-compose restart
```

要更改Harbour的配置，请先停止现有的Harbour实例并更新harbor.cfg。然后运行prepare脚本来填充配置。最后重新创建并启动Harbour的实例：

```
docker-compose down -v
vim harbor.cfg
./prepare
docker-compose up -d
```

配置Harbour在自定义端口上侦听

默认情况下，Harbour侦听端口80（HTTP）和443（HTTPS，如果已配置）同时用于管理界面和docker命令，则可以使用定制的命令对其进行配置。

修改docker-compose.yml文件，将里面的端口替换为自定义的端口，例如8888：80