使用 Harbor 搭建 Docker 私有仓库

Name：曲中岭

Email:zlinggu@126.com

Q Q :441869115

第一章 部署准备

1.1 目的

搭建自己的 docker 私有仓库，并可以通过 web 管理。

1.2 规划

OS	：CentOS_7.5 x64
IP	：172.16.6.31-32
docker-ce-cli	：18.09.0
docker-ce	：18.09.0
docker-compose	：1.23.1
Harbor	：1.5.4

其中 172.16.6.32 部署私有仓库，172.16.6.31 打包、上传、下载镜像等。

这里使用 harbor 搭建私有仓库，harbor 是 vmware 开源出来的，项目地址 https://github.com/goharbor/harbor

当然也可以使用官方提供的镜像进行搭建，只是没有认证、web 等好多个性化功能。

第二章 docker 安装

操作对象：

172.16.6.31

172.16.6.32

安装方法有很多，这里选择其中一种，rpm 方式。

2.1 安装

添加 docker 源：

```
yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
```

或者使用国内阿里云或者清华的源：

```
yum-config-manager --add-repo  https://mirrors.aliyun.com/docker-
ce/linux/centos/docker-ce.repo

yum-config-manager --add-repo
https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/centos/docker-
ce.repo
```

从指定源安装 docker-ce ：

```
yum install docker-ce --enablerepo=docker-ce-stable -y

systemctl start docker
systemctl enable docker
```

查看是否开机运行：

```
systemctl list-unit-files|grep docker
```

2.2 确认

```
docker version
```

```
[root@docker01 ~]# docker version
Client:
 Version:           18.09.0
 API version:       1.39
 Go version:        go1.10.4
 Git commit:        4d60db4
 Built:             Wed Nov  7 00:48:22 2018
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.0
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.4
  Git commit:       4d60db4
  Built:            Wed Nov  7 00:19:08 2018
  OS/Arch:          linux/amd64
  Experimental:     false
```

2.3 ubuntu 安装（补充）

方法有很多，这里只说一种。

```
curl -sSL https://get.docker.com/ | sh
service start docker
sysv-rc-conf --list|grep docker
update-rc.d  docker  start 90 3 4 5 . stop 20 0 1 2 6 .
sysv-rc-conf --list|grep docker
docker version
```

第三章 docker-compose 安装

操作对象:

172.16.6.32

harbor 的几个组件是用 docker-compose 启动和管理的，所以首先安装 docker-compose。

```
curl -L
https://github.com/docker/compose/releases/download/1.23.1/docker-
compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
docker-compose version
```

```
[root@docker02 harbor]# docker-compose version
docker-compose version 1.23.1, build b02f1306
docker-py version: 3.5.0
CPython version: 3.6.7
OpenSSL version: OpenSSL 1.1.0f  25 May 2017
[root@docker02 harbor]#
```

第四章 harbor 安装

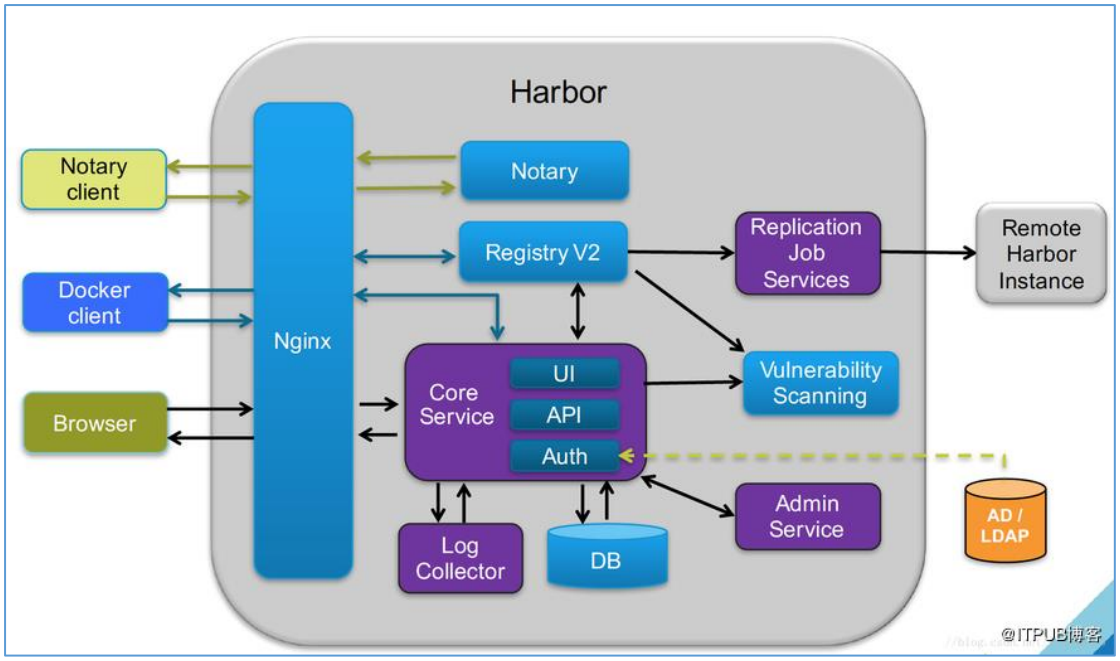
操作对象：

172.16.6.32

harbor 项目在 github 上开源，地址：https://github.com/goharbor/harbor

安装方式有多种，这里使用离线形式安装。

4.1 harbor 架构



- Proxy：对应启动组件 nginx。它是一个 nginx 反向代理，代理 Notary client（镜像认证）、Docker client（镜像上传下载等）和浏览器的访问请求（Core Service）给后端的各服务；
- UI（Core Service）：对应启动组件 harbor-ui。底层数据存储使用 mysql 数据库，主要提供了四个子功能：
    - UI：一个 web 管理页面 ui；
    - API：Harbor 暴露的 API 服务；
    - Auth：用户认证服务，decode 后的 token 中的用户信息在这里进行认证；auth 后端可以接 db、ldap、uaa 三种认证实现；
    - Token 服务（上图中未体现）：负责根据用户在每个 project 中的 role 来为每一个 docker push/pull 命令 issuing 一个 token，如果从 docker client 发送给 registry 的请求没有带 token，registry 会重定向请求到 token 服务创建 token。
- Registry：对应启动组件 registry。负责存储镜像文件，和处理镜像的 pull/push 命令。Harbor 对镜像进行强制的访问控制，Registry 会将客户端的每个 pull、push 请求转发到 token 服务来获取有效的 token。

- Admin Service：对应启动组件 harbor-adminserver。是系统的配置管理中心附带检查存储用量，ui 和 jobserver 启动时候需要加载 adminserver 的配置；
- Job Sevice：对应启动组件 harbor-jobservice。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；
- Log Collector：对应启动组件 harbor-log。日志汇总组件，通过 docker 的 log-driver 把日志汇总到一起；
- Volnerability Scanning：对应启动组件 clair。负责镜像扫描
- Notary：对应启动组件 notary。负责镜像认证
- DB：对应启动组件 harbor-db，负责存储 project、 user、 role、replication、image_scan、access 等的 metadata 数据。

4.2 4.1 下载

到这里选择合适的版本：
https://github.com/goharbor/harbor/releases

我这里使用离线的形式安装，800 多 M，耐心等待，这里选择最新的 1.5.4 版本。

```
wget https://storage.googleapis.com/harbor-releases/harbor-offline-installer-v1.5.4.tgz
tar xf harbor-offline-installer-v1.5.4.tgz
cd harbor
```

也可以使用在线安装形式

```
wget https://storage.googleapis.com/harbor-releases/release-1.7.0/harbor-online-installer-v1.7.5.tgz
```

4.3 修改配置文件

4.3.1 文件 harbor.cfg

修改 harbor.cfg 中的内容

```
hostname = reg.mydomain.com
```

为：

```
hostname = 172.16.6.32
```

提示：
1、该文件中，不要保留#hostname = reg.mydomain.com，因为 install 中会使用相关关键字做环境检查
2、该文件还有很多其他配置，如有需要可根据需要进行配置，当然有些也可以再安装完成

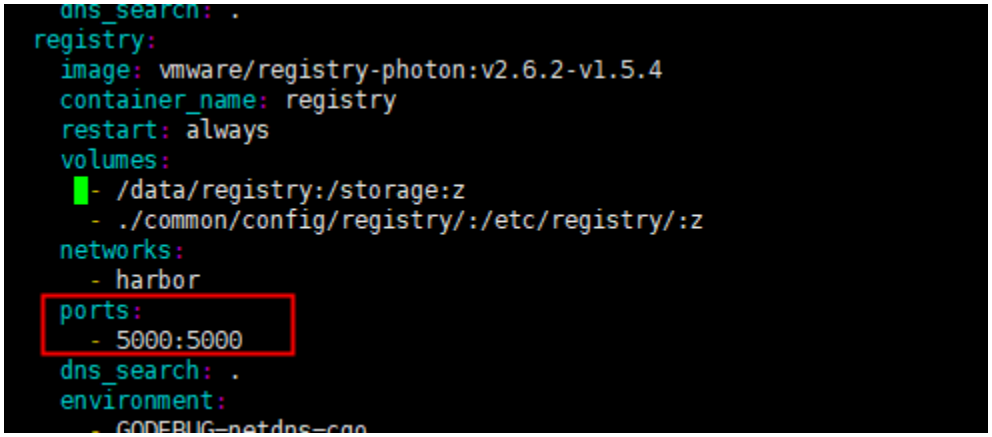
后再配置，比如邮件通知部分

4.3.2 文件 docker-compose.yml

该文件也是使用 yaml 语法，和 ansible-playbook 使用的是一样的。
在该文件的 registry 段落，增加如下两行

```
ports:
  - 5000:5000
```

如果不添加，也可以直接使用 80 进行访问。



重要提醒：
这类也可以不修改，不进行这一步。直接使用 80 端口，内置的 nginx 可以代理。

4.3.3 动态修改

如果 harbor 已经安装完成并且运行了一段时间，此时修改了配置文件，可使用如下步骤:

加载配置文件

```
./prepare
```

```
[root@harbor13201 harbor]#
[root@harbor13201 harbor]# ./prepare
Clearing the configuration file: ./common/config/adminserver/env
Clearing the configuration file: ./common/config/core/env
Clearing the configuration file: ./common/config/core/app.conf
Clearing the configuration file: ./common/config/core/private_key.pem
Clearing the configuration file: ./common/config/db/env
Clearing the configuration file: ./common/config/jobservice/env
Clearing the configuration file: ./common/config/jobservice/config.yml
Clearing the configuration file: ./common/config/registry/config.yml
Clearing the configuration file: ./common/config/registry/root.crt
Clearing the configuration file: ./common/config/registryctl/env
Clearing the configuration file: ./common/config/registryctl/config.yml
Clearing the configuration file: ./common/config/nginx/nginx.conf
Clearing the configuration file: ./common/config/log/logrotate.conf
loaded secret from file: /data/secretkey
Generated configuration file: ./common/config/nginx/nginx.conf
Generated configuration file: ./common/config/adminserver/env
Generated configuration file: ./common/config/core/env
Generated configuration file: ./common/config/registry/config.yml
Generated configuration file: ./common/config/db/env
Generated configuration file: ./common/config/jobservice/env
Generated configuration file: ./common/config/jobservice/config.yml
Generated configuration file: ./common/config/log/logrotate.conf
Generated configuration file: ./common/config/registryctl/env
Generated configuration file: ./common/config/core/app.conf
Generated certificate, key file: ./common/config/core/private_key.pem, cert file: ./common/config/registry/root.crt
The configuration files are ready, please use docker-compose to start the service.
[root@harbor13201 harbor]#
```

删除实例，会停止并删除所有相关的容器

```
docker-compose down
```

重新生成实列:

```
docker-compose up -d
```

4.3.4 使用 https

harbor.cfg 文件中，修改如下三项：

```
ui_url_protocol = https
ssl_cert = /data/cert/docker.dm-ai.cn.pem
ssl_cert_key = /data/cert/docker.dm-ai.cn.key
```

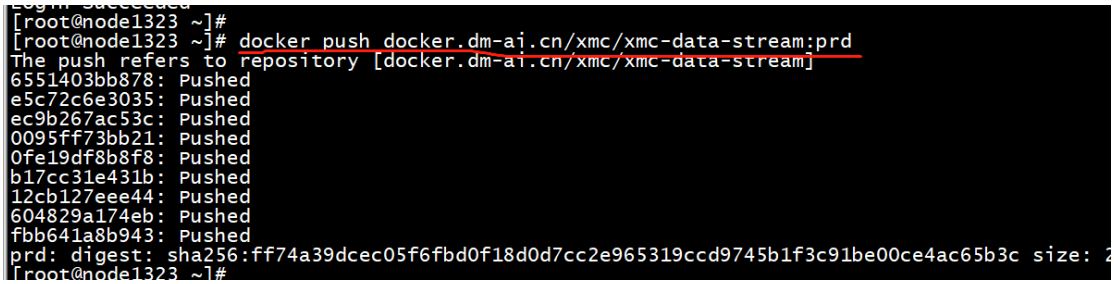
```
#It can be set to https if ssl is enabled on nginx.
ui_url_protocol = https

#Maximum number of job workers in job service
max_job_workers = 10

#Determine whether or not to generate certificate for the
#If the value is on, the prepare script creates new root
#for generating token to access the registry. If the valu
#This flag also controls the creation of the notary signe
customize_crt = on

#The path of cert and key files for nginx, they are appli
#ssl_cert = /data/cert/server.crt
#ssl_cert_key = /data/cert/server.key
ssl_cert = /data/cert/docker.dm-ai.cn.pem
ssl_cert_key = /data/cert/docker.dm-ai.cn.key

#The path of secretkey storage
```

证书和使用的 hostname 要一致

然后执行 4.3.3 步骤，即可开启 https。

使用了 https 之后，不需要进行 5.2 一步，可直接使用。

效果如下：



4.3.5 配置文件详解

https://blog.csdn.net/shenshouer/article/details/53390581

路径说明:

容器名	物理路径	容器路径
log	/var/log/harbor	/var/log/docker
registry 数据目录	/data/registry	/storage
registry 配置目录	./common/config/registry	/etc/registry
mysql 数据目	/data/database	/var/lib/mysql
mysql 变量文件	/common/config/db/env	
adminserver 配置目录	/data/config/	/etc/adminserver/config/
adminserver 秘钥文件	/data/secretkey	/etc/adminserver/key
adminserver 数据目录	/data/	/data/

容器名	物理路径	容器路径
UI 配置文件	./common/config/ui/app.conf	/etc/ui/app.conf
UI 私钥文件	./common/config/ui/private_key.pem	/etc/ui/private_key.pem
UI 秘钥文件	/data/secretkey	/etc/ui/key
UI	/data/ca_download/	/etc/ui/ca/
UI	/data/psc/	/etc/ui/token/
jobservice 日志文件	/data/job_logs	/var/log/jobs
jobservice 配置文件	./common/config/jobservice/app.conf	/etc/jobservice/app.conf
jobservice 秘钥文件	/data/secretkey	/etc/jobservice/key
proxy 配置文件	./common/config/nginx	/etc/nginx

4.4 安装

执行

```
./install.sh
```

等待安装完成即可。其实就是下载指定的几个镜像，并启动。

一共分 5 步（Setp 0-4）：

[Step 0]: checking installation environment

[Step 1]: loading Harbor images

[Step 2]: preparing environment

[Step 3]: checking existing instance of Harbor

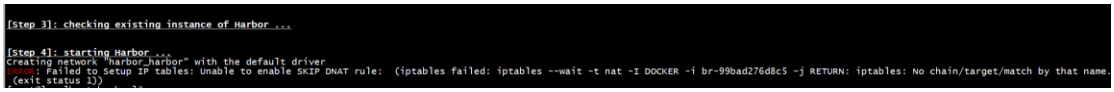
[Step 4]: starting Harbor

安装完成之后的提示：

```
[Step 4]: starting Harbor ...
Creating network "harbor_harbor" with the default driver
Creating harbor-log ... done
Creating redis              ... done
Creating harbor-adminserver ... done
Creating harbor-db          ... done
Creating registry           ... done
Creating harbor-ui          ... done
Creating harbor-jobservice  ... done
Creating nginx              ... done

✔ ----Harbor has been installed and started successfully.----

Now you should be able to visit the admin portal at http://172.16.6.32.
For more details, please visit https://github.com/vmware/harbor .
```

如果出现以下报错，原因是关闭防火墙之后 docker 需要重启，执行以下命令重启 docker 即可

service docker restart



4.5 启动和停止

启动 Harbor

```
docker-compose start
```

停止 Harbor

```
docker-comose stop
```

重启 Harbor

```
docker-compose restart
```

4.6 观察

下载好的镜像:

```
[root@docker02 harbor]# docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
vmware/redis-photon           v1.5.4              c7a3c332ff8f        3 weeks ago         210MB
vmware/clair-photon           v2.0.6-v1.5.4       faf3e2f1841f        3 weeks ago         302MB
vmware/notary-server-photon   v0.5.1-v1.5.4       f9a7e87fa884        3 weeks ago         209MB
vmware/notary-signer-photon   v0.5.1-v1.5.4       d9c8ddb0da72        3 weeks ago         207MB
vmware/registry-photon        v2.6.2-v1.5.4       6b4bdec4101e        3 weeks ago         196MB
vmware/nginx-photon           v1.5.4              ef99f7e61229        3 weeks ago         132MB
vmware/harbor-log             v1.5.4              13ae381fcfc5        3 weeks ago         198MB
vmware/harbor-jobservice      v1.5.4              aea9fd3c3fc0        3 weeks ago         192MB
vmware/harbor-ui              v1.5.4              a947b8b53a8f        3 weeks ago         209MB
vmware/harbor-adminserver     v1.5.4              d9ead0ee5c4a        3 weeks ago         181MB
vmware/harbor-db              v1.5.4              17cc3586bcd3        3 weeks ago         525MB
vmware/mariadb-photon         v1.5.4              446d2083018d        3 weeks ago         525MB
vmware/postgresql-photon      v1.5.4              0f4f752b7a90        3 weeks ago         219MB
photon                        1.0                 03c1901c3cd5        5 weeks ago         127MB
vmware/harbor-migrator        v1.5.0              466c57ab0dc3        6 months ago        1.16GB
[root@docker02 harbor]#
```

运行的容器：

```
[root@docker02 harbor]# docker ps
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS                    PORTS
                    NAMES
5eed15dbe6fc        vmware/nginx-photon:v1.5.4           "nginx -g 'daemon of…"   About a minute ago  Up About a minute (healthy)   0.0.0.0:80->80/tcp, 0.0.0.0:4
4443/tcp            nginx
f7b204d22419        vmware/harbor-jobservice:v1.5.4      "/harbor/start.sh"       About a minute ago  Up About a minute
                    harbor-jobservice
5dc098469a42        vmware/harbor-ui:v1.5.4              "/harbor/start.sh"       About a minute ago  Up About a minute (healthy)
                    harbor-ui
3b8dd4921621        vmware/harbor-db:v1.5.4              "/usr/local/bin/dock…"   About a minute ago  Up About a minute (healthy)   3306/tcp
                    harbor-db
5fe33981a0c2        vmware/harbor-adminserver:v1.5.4     "/harbor/start.sh"       About a minute ago  Up About a minute (healthy)
                    harbor-adminserver
a0cc324ff19b        vmware/registry-photon:v2.6.2-v1.5.4 "/entrypoint.sh serv…"   About a minute ago  Up About a minute (healthy)   5000/tcp
                    registry
a91b28e5fa6d        vmware/redis-photon:v1.5.4           "docker-entrypoint.s…"   About a minute ago  Up About a minute             6379/tcp
                    redis
01e337f8310a        vmware/harbor-log:v1.5.4             "/bin/sh -c /usr/loc…"   About a minute ago  Up About a minute (healthy)   127.0.0.1:1514->10514/tcp
                    harbor-log
```

端口监听：

```
[root@docker02 harbor]# netstat -tnlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:1514          0.0.0.0:*               LISTEN      10694/docker-proxy
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      967/sshd
tcp6       0      0 :::80                   :::*                    LISTEN      11382/docker-proxy
tcp6       0      0 :::22                   :::*                    LISTEN      967/sshd
tcp6       0      0 :::443                  :::*                    LISTEN      11361/docker-proxy
tcp6       0      0 :::4443                 :::*                    LISTEN      11334/docker-proxy
[root@docker02 harbor]#
```

/data 中有以下目录做了映射，是上述容器中映射出来的目录，至于如何映射的，可查看 docker-compose.yml 文件

```
[root@docker02 harbor]# ll /data/
总用量 32
drwxr-xr-x. 2   10000 10000 4096 11月 12 14:11 ca_download
drwxr-xr-x. 2   10000 10000 4096 11月 12 14:08 config
drwxr-xr-x. 5   10000 10000 4096 11月 12 14:11 database
drwxr-xr-x. 2   10000 10000 4096 11月 12 14:11 job_logs
drwxr-xr-x. 2   10000 10000 4096 11月 12 14:11 psc
drwxr-xr-x. 2 polkitd root  4096 11月 12 14:08 redis
drwxr-xr-x. 2   10000 10000 4096 11月 12 14:11 registry
-rw-------. 1   10000 10000   16 11月 12 14:07 secretkey
```

第五章 镜像操作

操作对象：

172.16.6.31

当然也可以操作其他已经安装了 docker 的机器，就像和 dockerhub 交互一样。

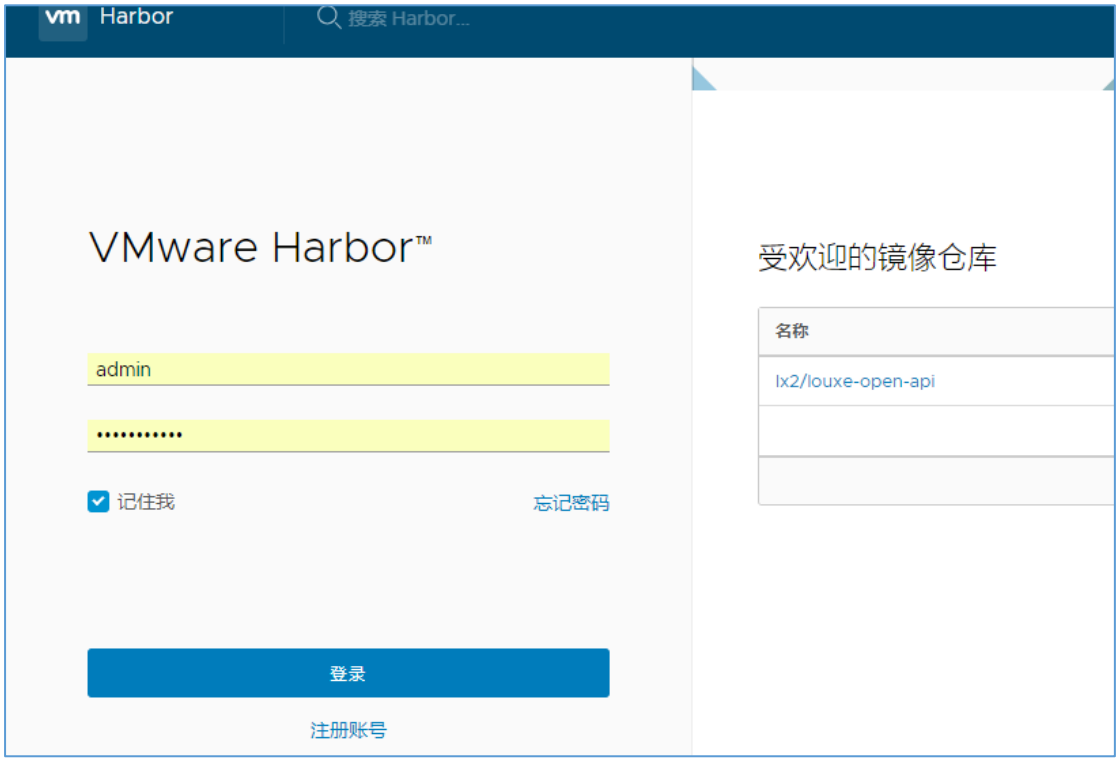
5.1 添加项目

使用浏览器打开：

http://172.16.6.32/harbor/sign-in

默认账号：admin

默认密码：Harbor12345



登陆后如下图创建项目，项目名叫做 lx2

注意，必须先添加项目，对于不存在的项目，无法推送，如下图，由于不存在 open 项目，所以无法推送。这是 harbor 相对于官方私有仓库的一个变化。

```
[root@docker01 ~]# docker push 172.16.6.32:5000/open/louxe-open-api
The push refers to repository [172.16.6.32:5000/open/louxe-open-api]
270f2c3d6b0b: Preparing
23885d16cf59: Preparing
35c20f26d188: Preparing
c3fe59dd9556: Preparing
6ed1a81ba5b6: Preparing
a3483ce177ce: Waiting
ce6c8756685b: Waiting
30339f20ced0: Waiting
0eb22bfb707d: Waiting
a2ae92ffcd29: Waiting
denied: requested access to the resource is denied
[root@docker01 ~]# ls .docker/
```

5.2 修改配置文件

由于 docker 默认使用 https，而我们的 harbor 是使用 http，所以要配置 docker 使用 http

修改

/etc/systemd/system/multi-user.target.wants/docker.service

或者

/lib/systemd/system/docker.service

文件中下面这行

```
#ExecStart=/usr/bin/dockerd -H unix://
```

为

```
ExecStart=/usr/bin/dockerd --insecure-registry 172.16.6.32:5000
```

如下图

```
[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd --insecure-registry 172.16.6.32:5000
#ExecStart=/usr/bin/dockerd -H unix://
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always
```

修改后使配置生效

```
systemctl daemon-reload
systemctl resart docker
```

如果不修改，会有如下报错，无法推送镜像到镜像库

```
[root@docker01 ~]# docker push 172.16.6.32:5000/lx2/louxe-open-api:v4
The push refers to repository [172.16.6.32:5000/lx2/louxe-open-api]
Get https://172.16.6.32:5000/v2/: http: server gave HTTP response to HTTPS client
[root@docker01 ~]#
[root@docker01 ~]#
```

或者使用如下方法，解决方法如下：

```
[root@docker01 system]# cat /etc/docker/daemon.json
{ "insecure-registries":["172.16.6.32:5000"]}
```

```
systemctl reload docker
service docker relod
```

使配置生效。

也可以使用如下命令，动态加载配置

```
kill -HUP $(pidof dockerd)
```

提醒：如果系统服务使用 init 管理的，配置文件的修改方式参考 6.2

5.3 登陆仓库

使用如下命令登陆仓库

```
docker login -u admin -p Harbor12345 172.16.6.32:5000
```

也可以使用

```
docker login 172.16.6.32:5000
```

根据提示输入账号密码。

登录的账号密码信息保存在以下文件中： ~/.docker/config.json

内容类似于如下，账号密码信息被加密了。可以拷贝这个文件到其他机器复用。

```
[root@master2 ~]# cat .docker/config.json
{
        "auths": {
                "registry-vpc.cn-shenzhen.aliyuncs.com": {
                        "auth": "c3lzdWhjcGxhYjpkbWFpQDIwMTg="
                },
                "registry.us-west-1.aliyuncs.com": {
                        "auth": "c3lzdWhjcGxhYjpkbWFpQDIwMTg="
                }
        }
}[root@master2 ~]#
```

如果不登陆会有如下提示，无法上传，也无法拉取

```
[root@docker01 .docker]# docker push 172.16.6.32:5000/lx2/louxe-open-api:v2
The push refers to repository [172.16.6.32:5000/lx2/louxe-open-api]
270f2c3d6b0b: Layer already exists
23885d16cf59: Layer already exists
35c20f26d188: Layer already exists
c3fe59dd9556: Layer already exists
6ed1a81ba5b6: Layer already exists
a3483ce177ce: Layer already exists
ce6c8756685b: Layer already exists
30339f20ced0: Layer already exists
0eb22bfb707d: Layer already exists
a2ae92ffcd29: Layer already exists
errors:
denied: requested access to the resource is denied
unauthorized: authentication required
```

登陆信息会保存在:

~/.docker/config.json

内容如下:

```
[root@docker01 ~]# cat .docker/config.json
{
        "auths": {
                "172.16.6.32:5000": {
                        "auth": "YWRtaW46SGFyYm9yMTIzNDU="
                }
        },
        "HttpHeaders": {
                "User-Agent": "Docker-Client/18.09.0 (linux)"
        }
}[root@docker01 ~]#
```

5.4 推送镜像

给已有镜像打标签，并推送

```
[root@docker01 ~]# docker images louxe-open-api
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
louxe-open-api      v2                  a937eef2d4df        3 days ago          771MB
[root@docker01 ~]#
```

```
docker tag a937eef2d4df 172.16.6.32:5000/lx2/louxe-open-api:v4
docker push 172.16.6.32:5000/lx2/louxe-open-api:v4
```

```
[root@docker01 ~]# docker push 172.16.6.32:5000/lx2/louxe-open-api:v4
The push refers to repository [172.16.6.32:5000/lx2/louxe-open-api]
270f2c3d6b0b: Layer already exists
23885d16cf59: Layer already exists
35c20f26d188: Layer already exists
c3fe59dd9556: Layer already exists
6ed1a81ba5b6: Layer already exists
a3483ce177ce: Layer already exists
ce6c8756685b: Layer already exists
30339f20ced0: Layer already exists
0eb22bfb707d: Layer already exists
a2ae92ffcd29: Layer already exists
v4: digest: sha256:631404e2988e05e44a986ac405809ec40c48f5d2c346dbbafd96ed3ea5b71623 size: 2424
```

注意，使用 harbor 搭建的私有镜像库，待推送的镜像标签中必须带有项目名，如上面的 lx2，这是一种经过增强的仓库。如果使用官方的，则有没有均可。

5.5 拉取镜像

使用如下命令拉取即可

```
docker pull 172.16.6.32:5000/lx2/louxe-open-api:v2
```

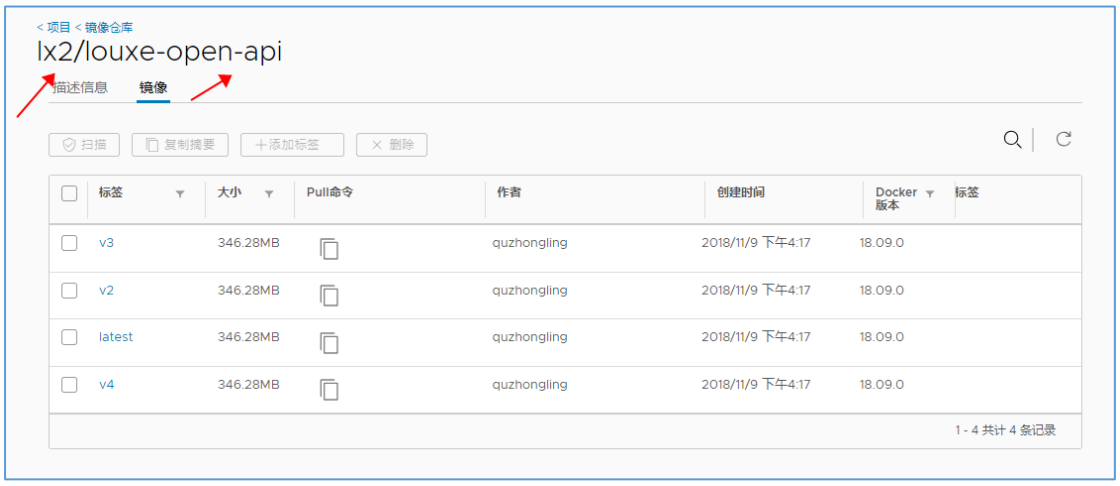
```
[root@docker01 ~]# docker images 172.16.6.32:5000/lx2/louxe-open-api:v4
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# docker pull 172.16.6.32:5000/lx2/louxe-open-api:v4
v4: Pulling from lx2/louxe-open-api
Digest: sha256:631404e2988e05e44a986ac405809ec40c48f5d2c346dbbafd96ed3ea5b71623
Status: Downloaded newer image for 172.16.6.32:5000/lx2/louxe-open-api:v4
[root@docker01 ~]#
[root@docker01 ~]# docker images 172.16.6.32:5000/lx2/louxe-open-api:v4
REPOSITORY                              TAG                 IMAGE ID            CREATED             SIZE
172.16.6.32:5000/lx2/louxe-open-api   v4                  a937eef2d4df        3 days ago          771MB
[root@docker01 ~]#
[root@docker01 ~]#
```

5.6 观察镜像库

使用 http://172.16.6.32/harbor/users 登陆

账号：admin

密码：Harbor12345



harbor 私有库可以实现镜像复制、镜像删除等操作，这里不再演示。

第六章 用官方的私有仓库(补充)

6.1 安装

```
docker run -d -v /data/docker-registry:/var/lib/registry -p 5000:5000
--restart=always --privileged=true --name registry registry:latest
```

注释:
-v /data/docker-registry:/var/lib/registry 本地的/data/docker-registry 映射到容器内
--restart=always 容器退出时总是重启容器
--name registry 指定容器的名称

6.2 调整配置

操作使出现如下错误:

```
[root@docker01 system]# docker push 172.16.6.32:5000/centos:7
The push refers to repository [172.16.6.32:5000/centos]
Get https://172.16.6.32:5000/v2/: http: server gave HTTP response to HTTPS client
[root@docker01 system]#
```

因为 docker 默认使用 https，私有仓库不是使用 https 的，解决方法如下:

```
[root@docker01 system]# cat /etc/docker/daemon.json
{ "insecure-registries":["172.16.6.32:5000"]}
```

```
systemctl reload docker
service docker relod
```

使配置生效。
也可以使用如下命令，动态加载配置

```
kill -HUP $(pidof dockerd)
```

6.3 相关操作

推送、拉取等都不需要认证，也不需要在打标签时指定项目名，如下

```
docker tag louxe-open-api 172.16.6.32:5000/louxe-open-api
docker push 172.16.6.32:5000/louxe-open-api
```

```
[root@docker01 system]# docker push 172.16.6.32:5000/louxe-open-api
The push refers to repository [172.16.6.32:5000/louxe-open-api]
270f2c3d6b0b: Pushed
23885d16cf59: Pushed
35c20f26d188: Pushed
c3fe59dd9556: Pushed
6ed1a81ba5b6: Pushed
a3483ce177ce: Pushed
ce6c8756685b: Pushed
30339f20ced0: Pushed
0eb22bfb707d: Pushed
a2ae92ffcd29: Pushed
latest: digest: sha256:631404e2988e05e44a986ac405809ec40c48f5d2c346dbbafd96ed3ea5b71623 size: 2424
```

查看私有仓库中有哪些镜像:

curl http://172.16.6.32:5000/v2/_catalog

查看某个镜像有哪些标签：

curl http://172.16.6.32:5000/v2/louxe-open-api/tags/list

```
[root@docker01 ~]# curl http://172.16.6.32:5000/v2/_catalog
{"repositories":["centos","louxe-open-api"]}
[root@docker01 ~]#
[root@docker01 ~]#
[root@docker01 ~]# curl http://172.16.6.32:5000/v2/louxe-open-api/tags/list
{"name":"louxe-open-api","tags":["v3","latest"]}
[root@docker01 ~]#
[root@docker01 ~]# curl http://172.16.6.32:5000/v2/centos/tags/list
{"name":"centos","tags":["7"]}
[root@docker01 ~]#
```

第七章 角色管理

基于角色的访问控制，RBAC，这个是 k8s 1.6 以后才加入的功能，harbor 在设计的开始就考虑进去，用户分为三种角色：项目管理员（MDRWS）、开发人员（RWS）和访客（RS），当然还有一个最高管理员权限 admin 系统管理员。

上面的简称大概说一下，M:管理、D:删除、R:读取、W:写入、S:查询，非常细致的权限管理体系。当然一个用户可以在不同的项目里面扮演不同角色，这个和现实的用户管理体系非常吻合。

第八章 开启 ldap 认证

8.1 修改配置文件

vim harbor.cfg
根据实际情况，修改如下六行信息

```
#Set it to ldap_auth if you want to verify a user's credentials against an LDAP ser
#auth_mode = db_auth
auth_mode = ldap_auth

#The url for an ldap endpoint.
#ldap_url = ldaps://ldap.mydomain.com
ldap_url = ldap://192.168.3.157:389

#A user's DN who has the permission to search the LDAP/AD server.
#If your LDAP/AD server does not support anonymous search, you should configure thi
#ldap_searchdn = uid=searchuser,ou=people,dc=mydomain,dc=com
ldap_searchdn = cn=app_bind,ou=apps,dc=dmai,dc=com

#the password of the ldap_searchdn
#ldap_search_pwd = password
ldap_search_pwd = 1234qwer

#The base DN from which to look up a user in LDAP/AD
#ldap_basedn = ou=people,dc=mydomain,dc=com
ldap_basedn = ou=people,dc=dmai,dc=com

#Search filter for LDAP/AD, make sure the syntax of the filter is correct.
#ldap_filter = (objectClass=person)

# The attribute used in a search to match a user, it could be uid, cn, email, sAMAc
ldap_uid = cn

#the scope to search for users, 0-LDAP_SCOPE_BASE, 1-LDAP_SCOPE_ONELEVEL, 2-LDAP_SC
ldap_scope = 2

#Timeout (in seconds)  when connecting to an LDAP Server. The default value (and mo
ldap_timeout = 5

#Verify certificate from LDAP server
ldap_verify_cert = true

#The base dn from which to lookup a group in LDAP/AD
#ldap_group_basedn = ou=group,dc=mydomain,dc=com
ldap_group_basedn = ou=group,dc=dmai,dc=com

#filter to search LDAP/AD group
ldap_group_filter = objectclass=group

#The attribute used to name a LDAP/AD group, it could be cn, name
ldap_group_gid = cn
```

auth_mode = ldap_auth
ldap_url = ldap://192.168.3.157:389
ldap_searchdn = cn=app_bind,ou=apps,dc=dmai,dc=com
ldap_search_pwd = 1234qwer
ldap_basedn = ou=people,dc=dmai,dc=com
ldap_group_basedn = ou=group,dc=dmai,dc=com

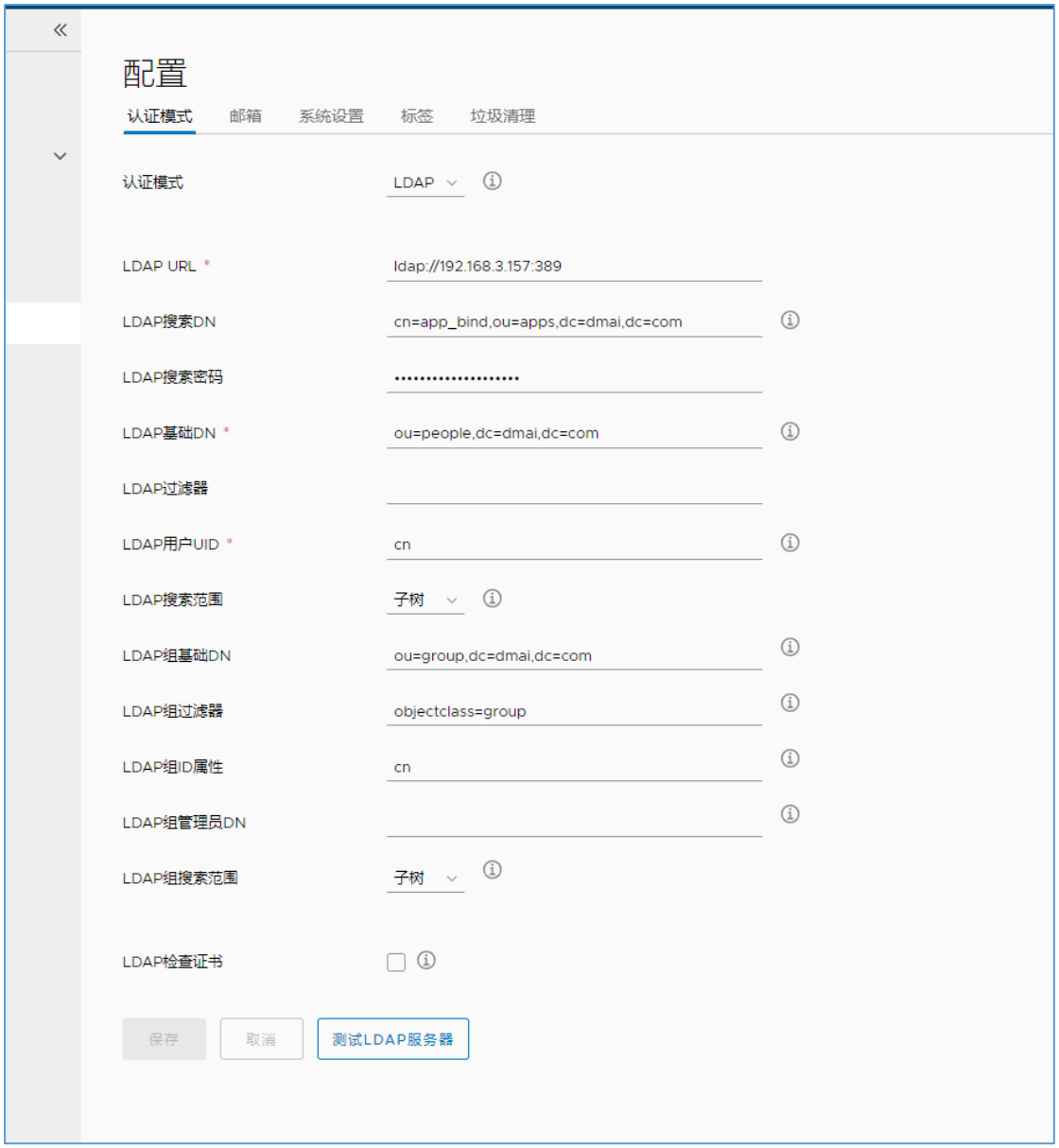
8.2 配置生效

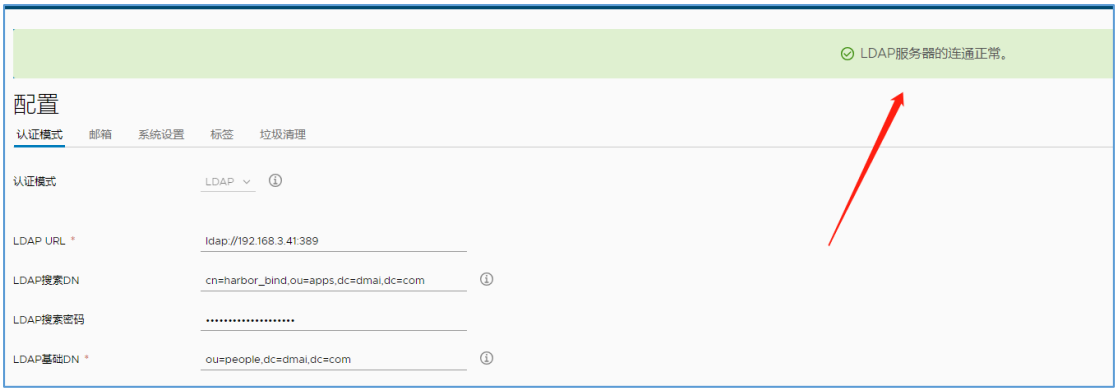
使用如下命令使配置生效

./install.sh
使配置生效

8.3 修改认证模式

首次配置之后，这里会生效，变成如下。也可以在这里手动填写，写好后，点击保存，并点击测试，验证是否通过




LDAP服务器的连通正常。
配置
认证模式
邮箱
系统设置
标签
垃圾清理
认证模式
LDAP
LDAP URL *
ldap://192.168.3.41:389
LDAP搜索DN
cn=harbor_bind,ou=apps,dc=dmai,dc=com
LDAP搜索密码
LDAP基础DN *
ou=people,dc=dmai,dc=com

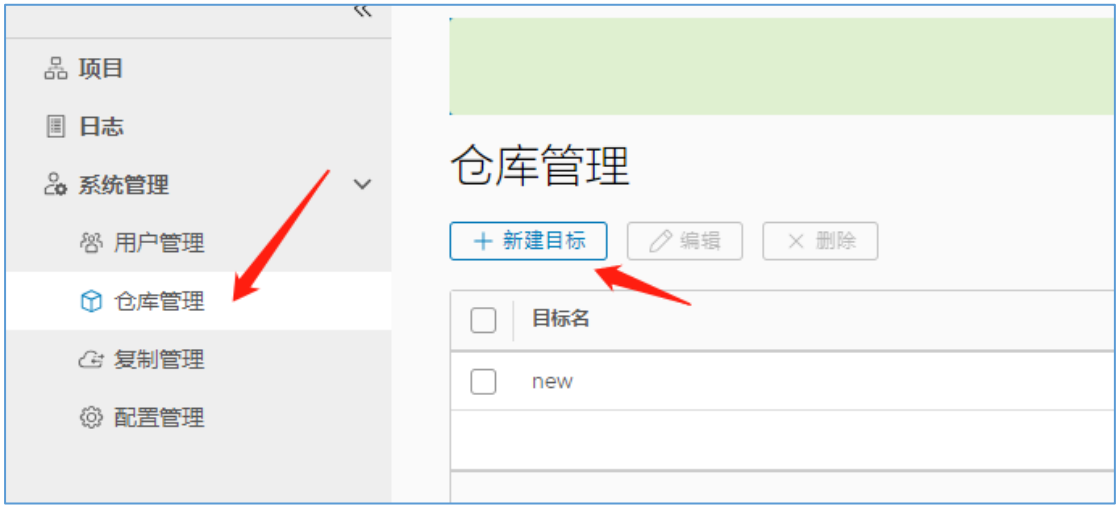
第九章 镜像复制功能

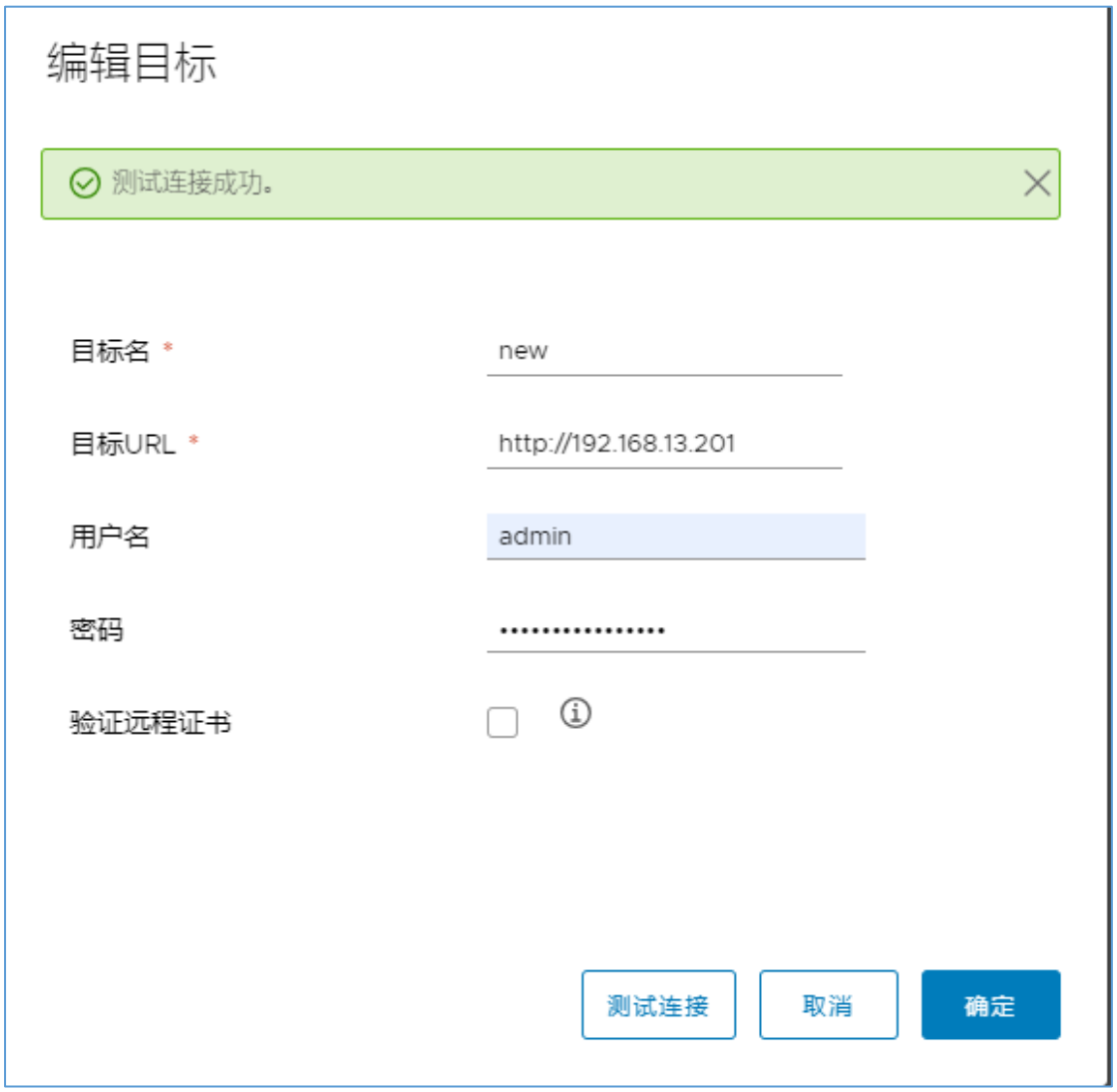
9.1 环境

A：v1.7.4，使用 http，192.168.3.201,hostname:192.168.3.201，认证模式 db_auth
B：v1.7.5，使用 http, 192.168.13.201,hostname:192.168.13.201，认证模式 db_auth

9.2 添加目标

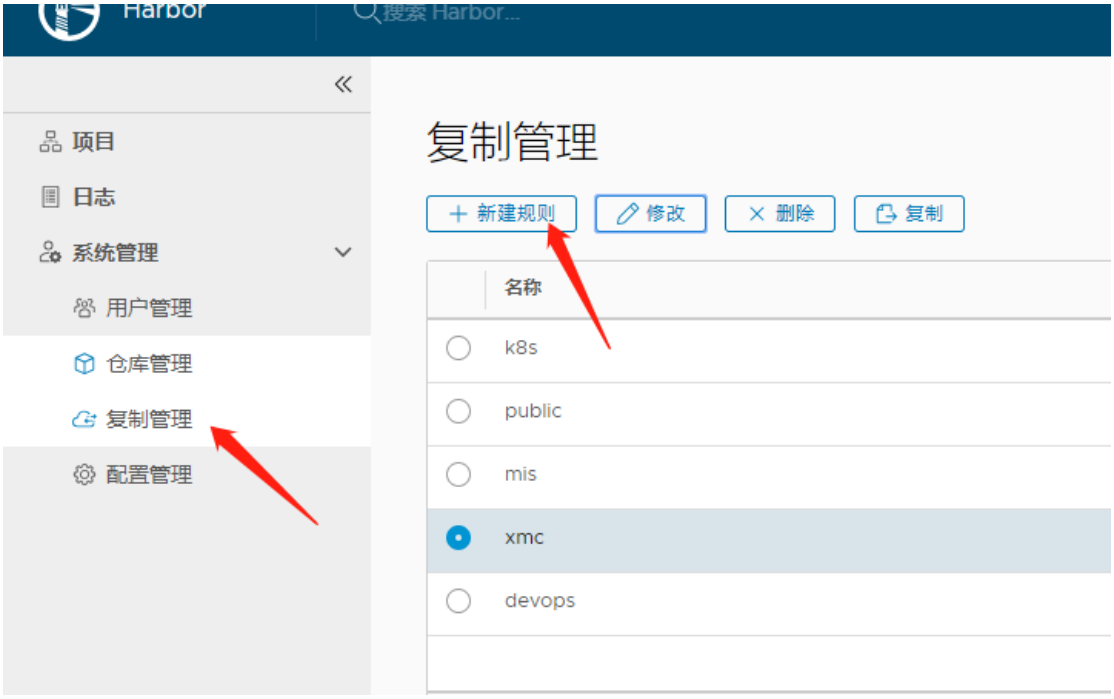
在 A 上操作

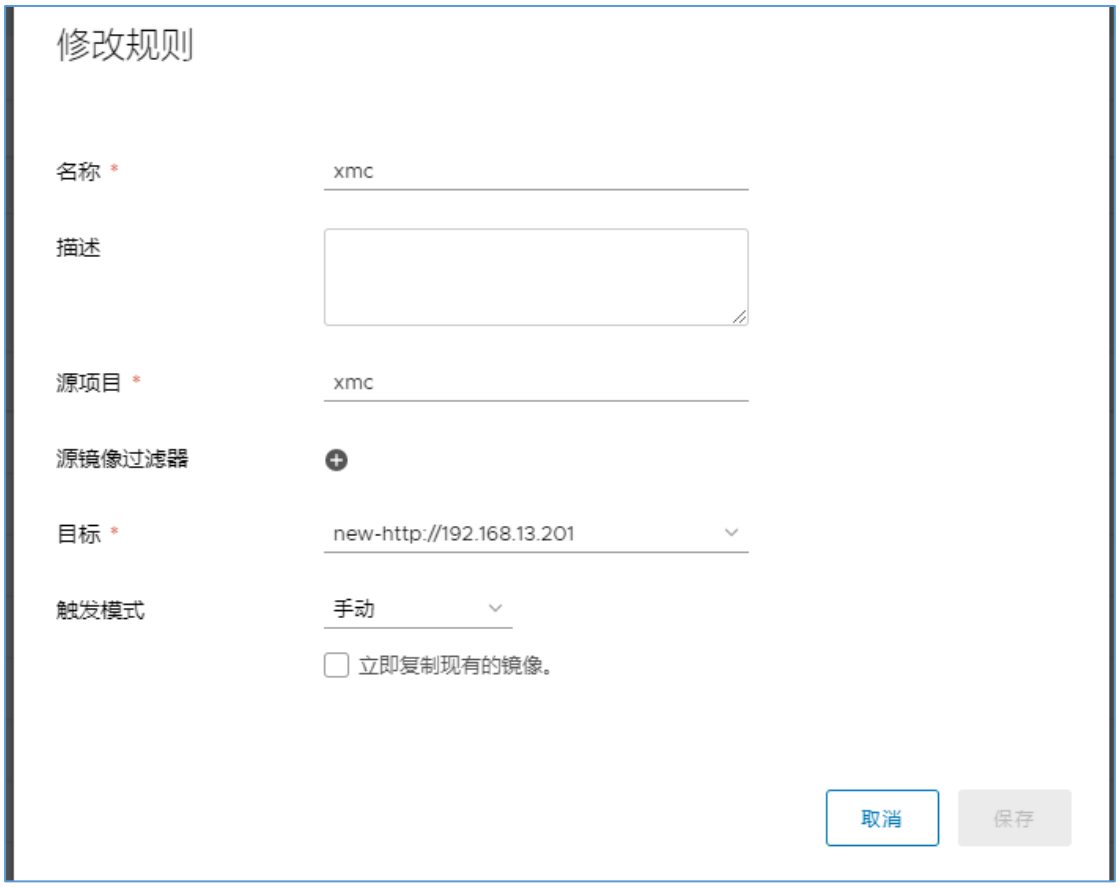




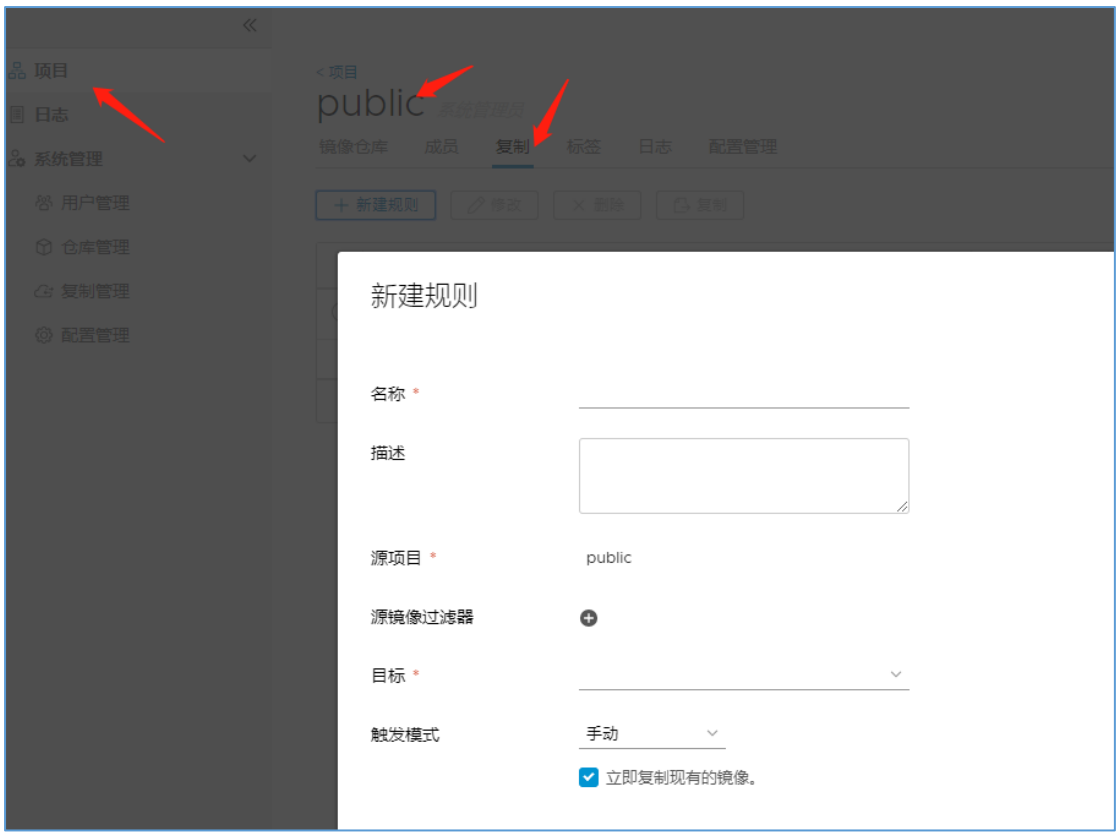
注意，这里的 url，一定要和 B 的 hostname 一致，否则是无法成功的

9.3 复制管理

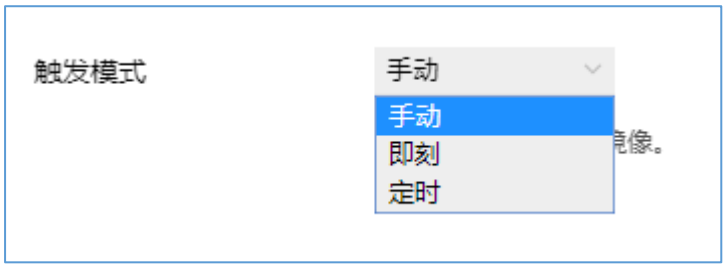




也可进入某个项目创建复制规则，这时源项目是无法修改的，就是当前项目

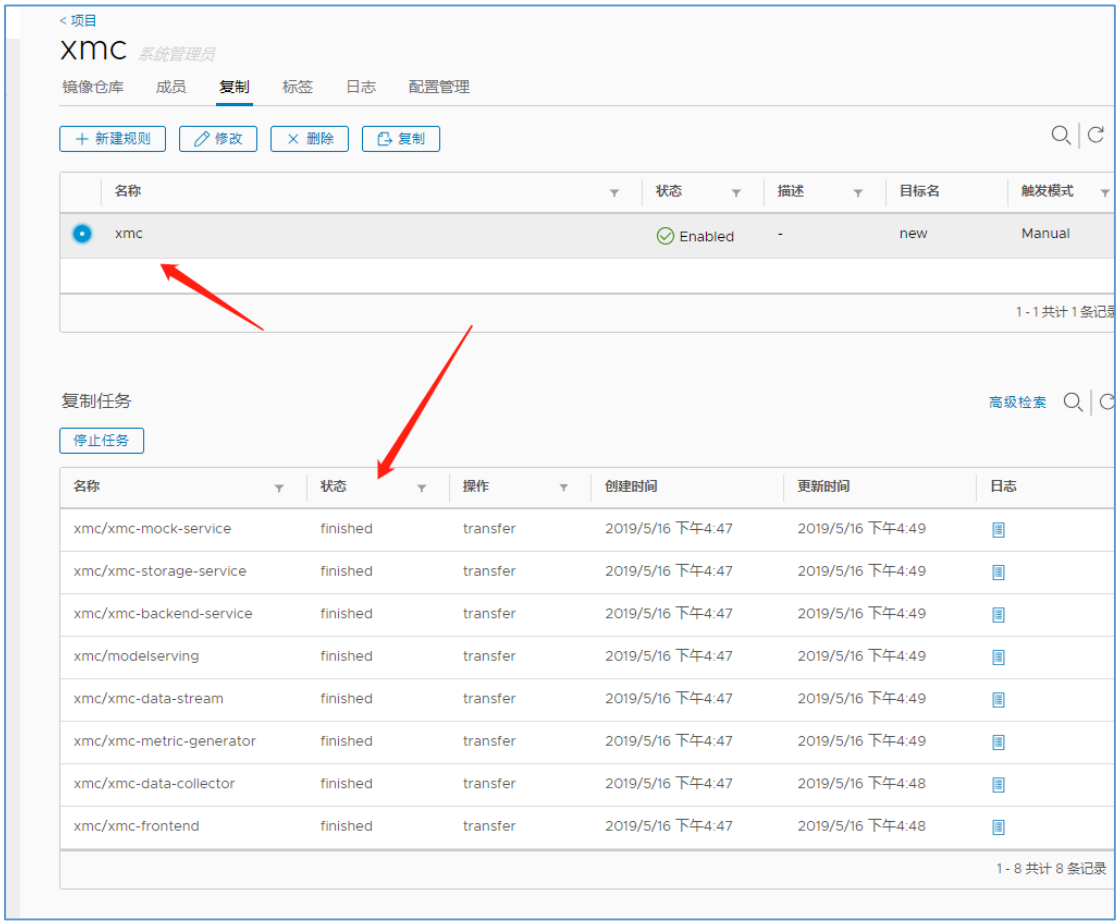


触发模式有三种，根据需要进行选择



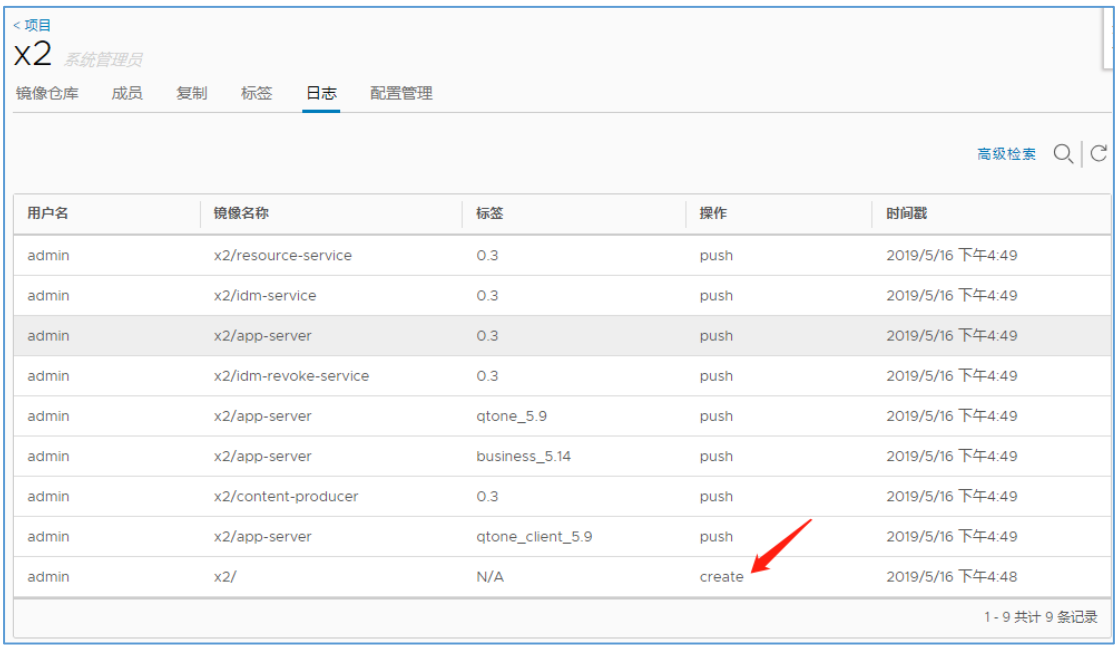
9.4 复制状态

可以查看某个复制规则的复制状态，如下图



9.5 注意

1、第一次复制，B 上无需创建项目，复制时会自动创建，如下图，同步时先创建项目



2、 新版的规则中，A 中删除了某个镜像或者标签，B 中也不会删除

第十章 k8s 变更

当仓库启用了认证后，需要进行如下操作:

9.1 添加 secret 资源

每个 ns 里面都要添加。

kubectl create secret docker-registry regsecret --namespace=xmc --docker-server=https://docker.dm-ai.cn --docker-username=admin --docker-password=*** --docker-email=***

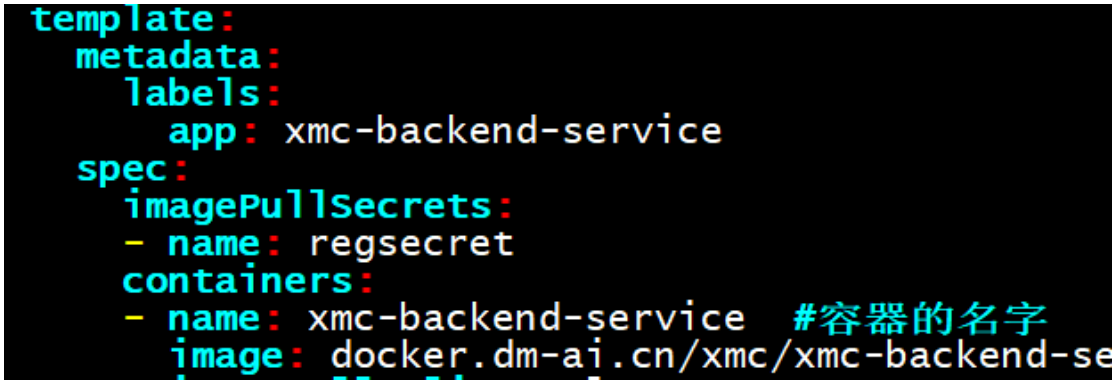
regsecret: 指定密钥的键名称，可自行定义
--docker-server: 指定 docker 仓库地址
--docker-username: 指定 docker 仓库账号
--docker-password: 指定 docker 仓库密码
--docker-email: 指定邮件地址(选填)

添加之后，效果如下

```
[root@master ~]#
[root@master ~]# kubectl get secret --all-namespaces|grep regsecret
devops         regsecret                                              kubernetes.io/dockercon
istio-system   regsecret                                              kubernetes.io/dockercon
quantong       regsecret                                              kubernetes.io/dockercon
x2-public      regsecret                                              kubernetes.io/dockercon
x2             regsecret                                              kubernetes.io/dockercon
xmc            regsecret                                              kubernetes.io/dockercon
[root@master ~]#
[root@master ~]#
```

9.2 yaml 文件中添加如下内容：

imagePullSecrets:
- name: regsecret



注意位置，和 containers 属于同一级别