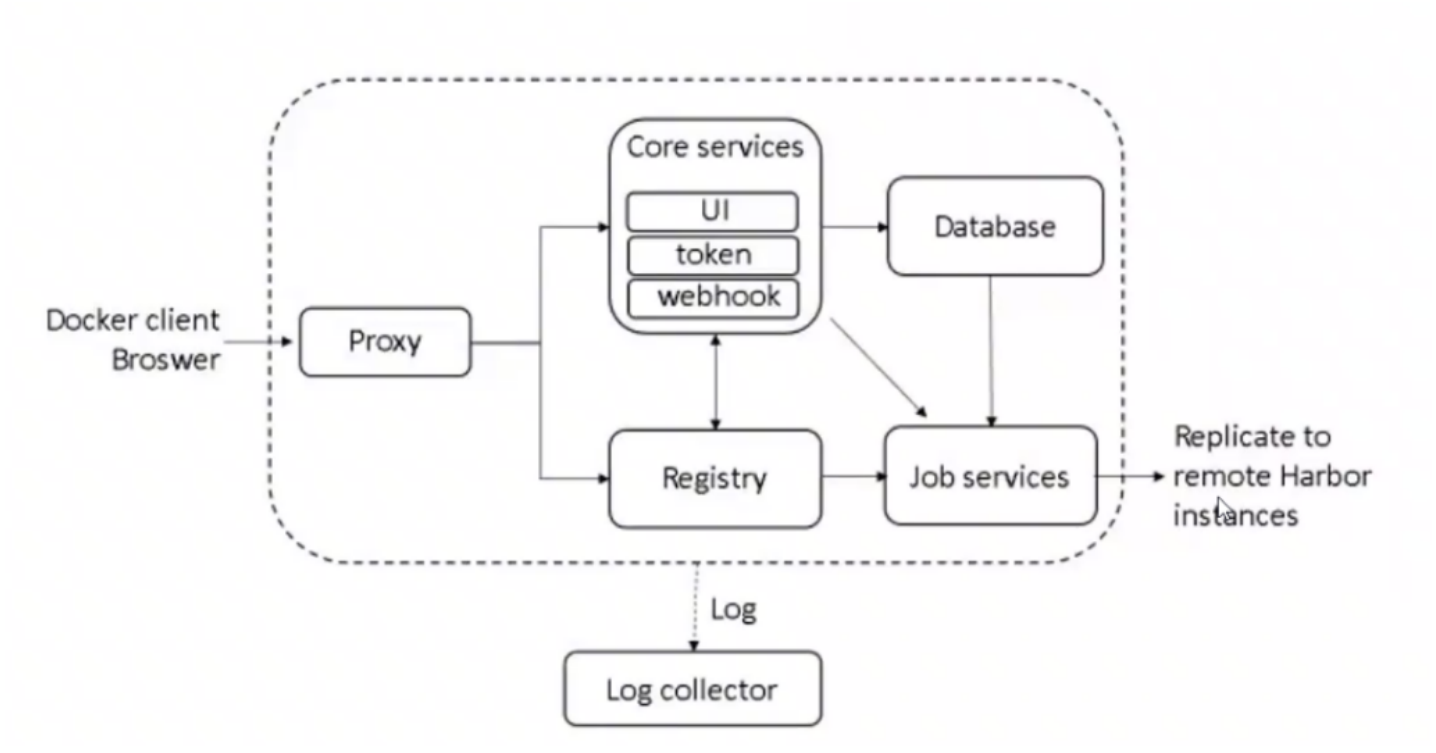
Harbor

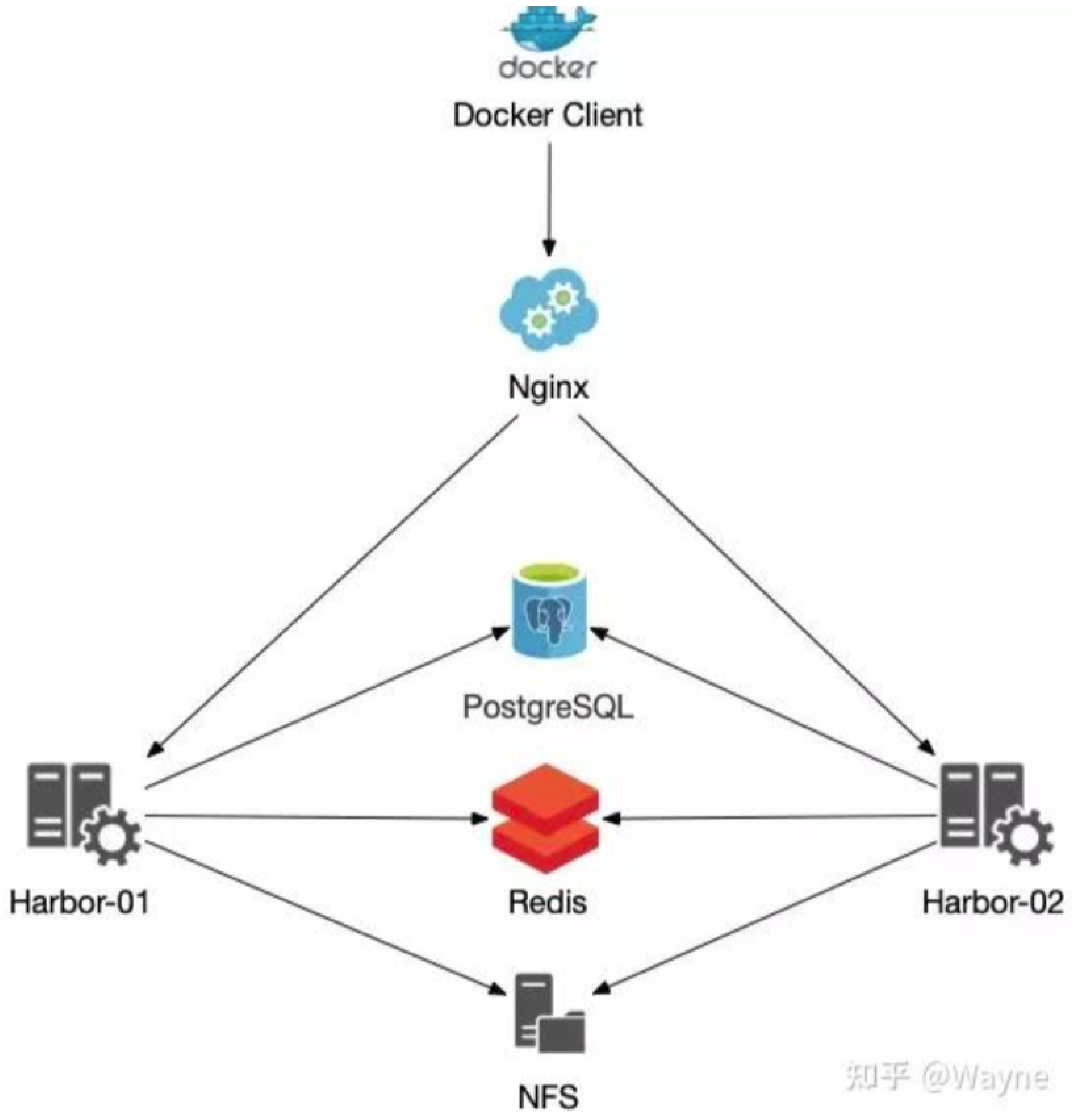
Harbor架构

Harbor的构成:

Harbor 在架构上主要有 Proxy、Registry、Core services、Database（Harbor-db）、Log collector（Harbor-log）、Job services 六个组件。



高可用架构：多实例共享后端存储



部署安装

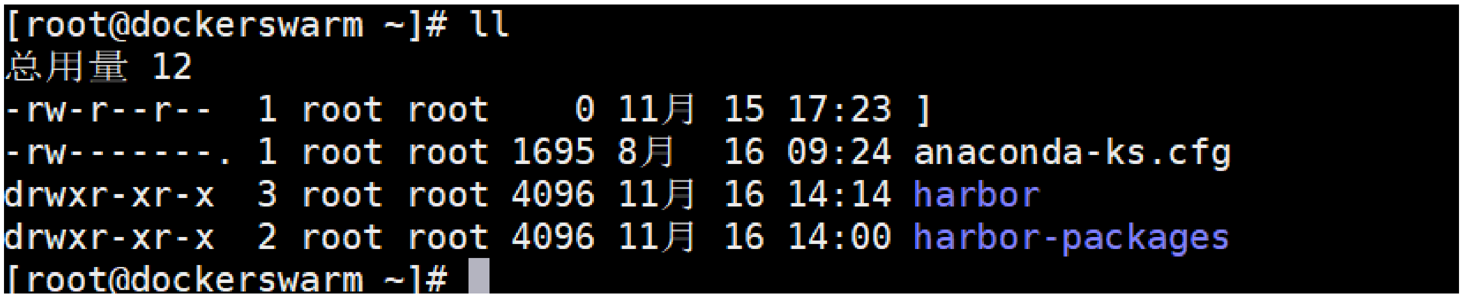
https://www.cnblogs.com/netcore3/p/17513138.html

下载harbor包

```
# Github源
wget https://github.com/goharbor/harbor/releases/download/v2.9.1/harbor-offline-installer-v2.9.1.tgz
# 代理源(推荐使用)
wget https://ghproxy.com/https://github.com/goharbor/harbor/releases/download/v2.5.3/harbor-offline-in
```

解压harbor文件包

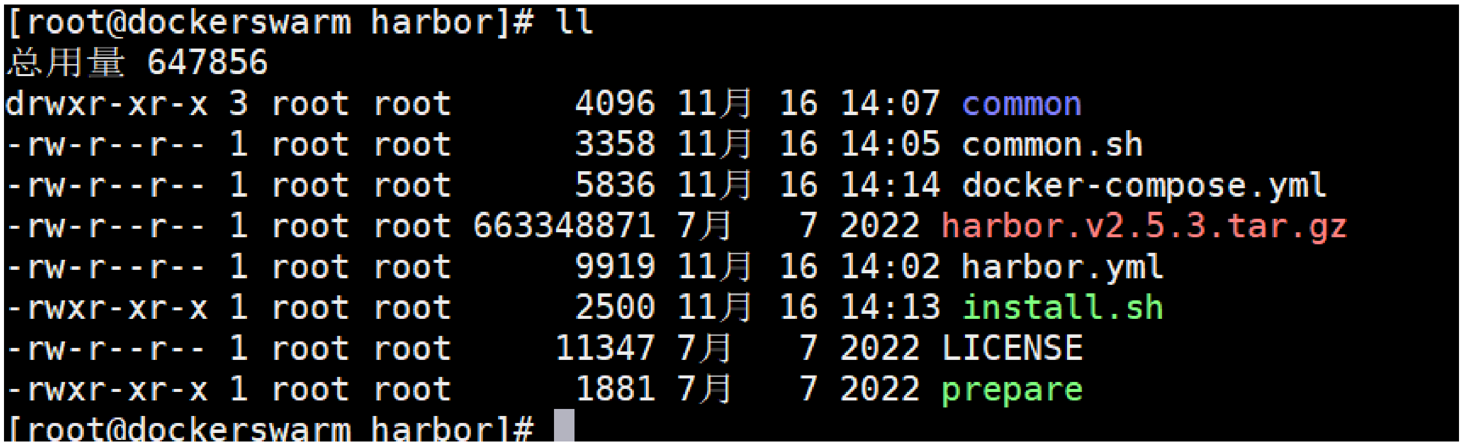
```
# 解压harbor文件
tar -zxvf harbor-offline-installer-v2.5.3.tgz
# 移动到 ~/harbor
mv harbor ~/harbor
```



```
[root@dockerswarm ~]# ll
总用量 12
-rw-r--r--  1 root root    0 11月  15 17:23 ]
-rw-------. 1 root root 1695 8月   16 09:24 anaconda-ks.cfg
drwxr-xr-x  3 root root 4096 11月  16 14:14 harbor
drwxr-xr-x  2 root root 4096 11月  16 14:00 harbor-packages
[root@dockerswarm ~]#
```

修改harbor.yml配置文件

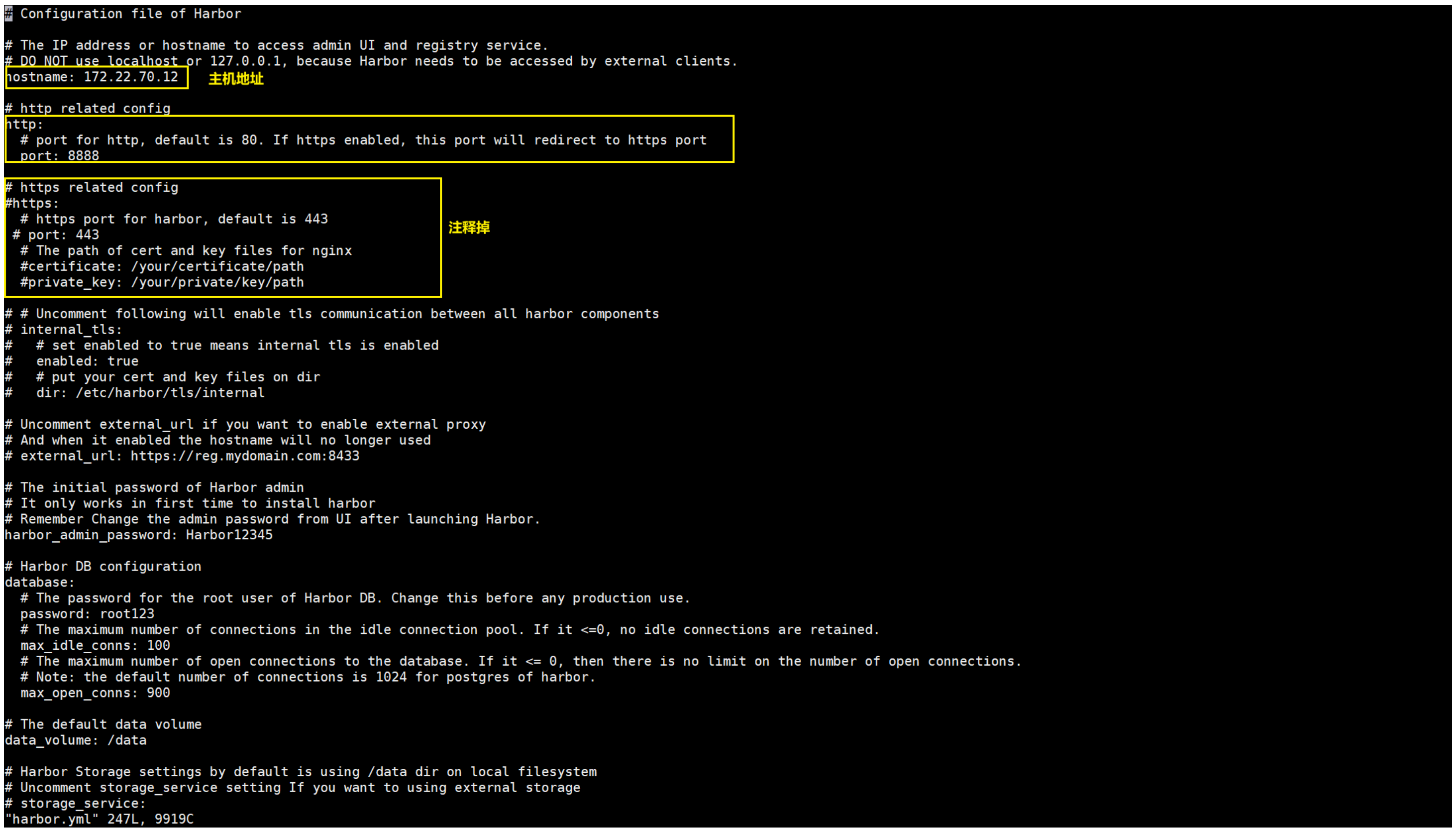
```
cd harbor
mv harbor.yml.tmpl harbor.yml
```



```
[root@dockerswarm harbor]# ll
总用量 647856
drwxr-xr-x 3 root root      4096 11月  16 14:07 common
-rw-r--r-- 1 root root      3358 11月  16 14:05 common.sh
-rw-r--r-- 1 root root      5836 11月  16 14:14 docker-compose.yml
-rw-r--r-- 1 root root 663348871 7月    7 2022 harbor.v2.5.3.tar.gz
-rw-r--r-- 1 root root      9919 11月  16 14:02 harbor.yml
-rwxr-xr-x 1 root root      2500 11月  16 14:13 install.sh
-rw-r--r-- 1 root root     11347 7月    7 2022 LICENSE
-rwxr-xr-x 1 root root      1881 7月    7 2022 prepare
[root@dockerswarm harbor]#
```

```
vi harbor.yml
```

修改主机地址，端口以及将https配置注释掉


```
# Configuration file of Harbor

# The IP address or hostname to access admin UI and registry service.
# DO NOT use localhost or 127.0.0.1, because Harbor needs to be accessed by external clients.
hostname: 172.22.70.12   主机地址

# http related config
http:
  # port for http, default is 80. If https enabled, this port will redirect to https port
  port: 8888

# https related config
#https:
  # https port for harbor, default is 443
 # port: 443
  # The path of cert and key files for nginx
  #certificate: /your/certificate/path
  #private_key: /your/private/key/path      注释掉

# # Uncomment following will enable tls communication between all harbor components
# internal_tls:
#   # set enabled to true means internal tls is enabled
#   enabled: true
#   # put your cert and key files on dir
#   dir: /etc/harbor/tls/internal

# Uncomment external_url if you want to enable external proxy
# And when it enabled the hostname will no longer used
# external_url: https://reg.mydomain.com:8433

# The initial password of Harbor admin
# It only works in first time to install harbor
# Remember Change the admin password from UI after launching Harbor.
harbor_admin_password: Harbor12345

# Harbor DB configuration
database:
  # The password for the root user of Harbor DB. Change this before any production use.
  password: root123
  # The maximum number of connections in the idle connection pool. If it <=0, no idle connections are retained.
  max_idle_conns: 100
  # The maximum number of open connections to the database. If it <= 0, then there is no limit on the number of open connections.
  # Note: the default number of connections is 1024 for postgres of harbor.
  max_open_conns: 900

# The default data volume
data_volume: /data

# Harbor Storage settings by default is using /data dir on local filesystem
# Uncomment storage_service setting If you want to using external storage
# storage_service:
"harbor.yml" 247L, 9919C
```

启动 harbor

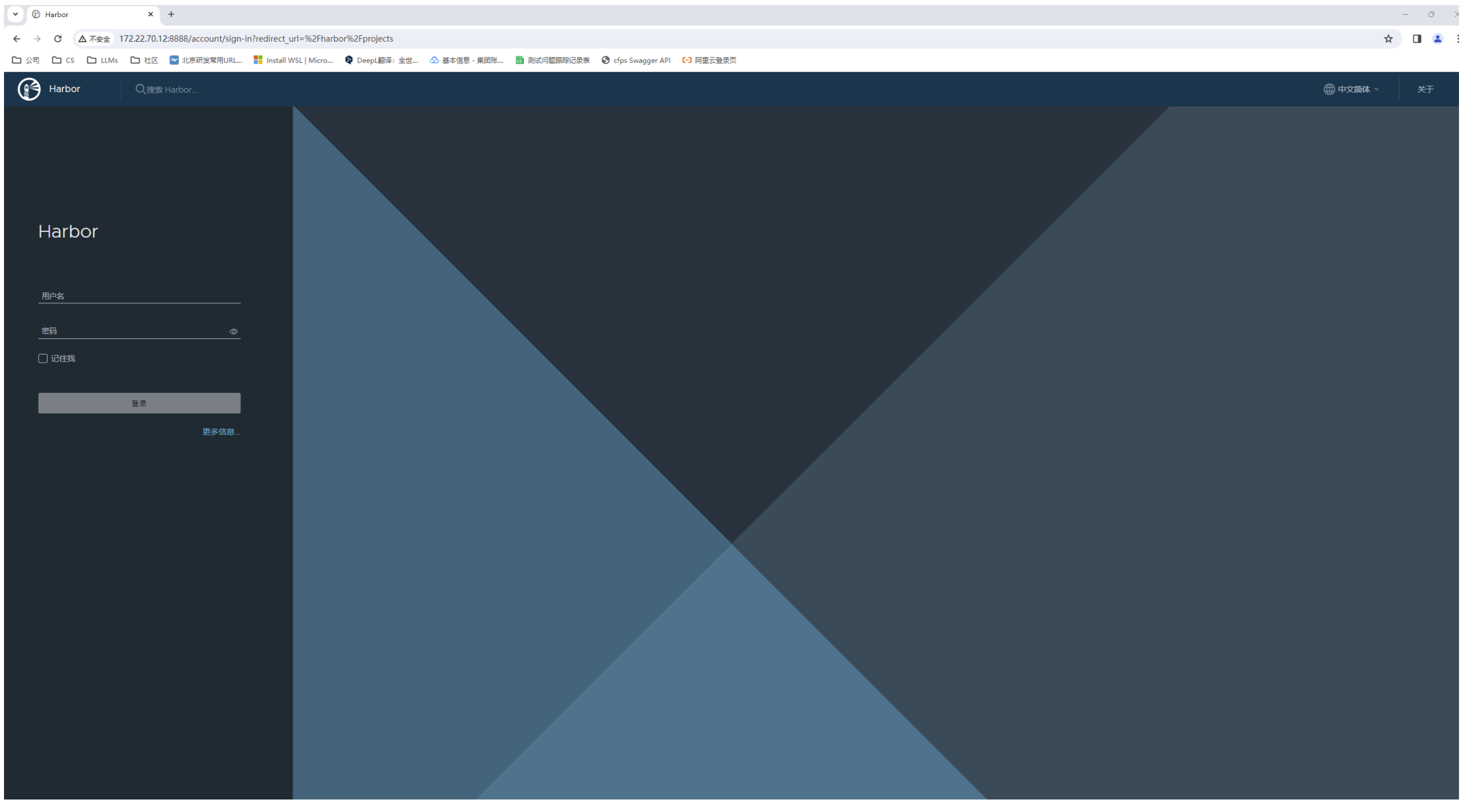
```
./install.sh
```

开启镜像扫描器(trivy)的启动方式:

```
# 开启trivy （默认安全扫描扫描器）
./install.sh --with-trivy
# 开启trivy 和 chartmuseum
./install.sh --with-trivy --with-chartmuseum
# 可选择其他镜像安全扫描器，默认为trivy
# Note: Please set hostname and other necessary attributes in harbor.yml first. DO NOT use localhost o
# Please set --with-notary if needs enable Notary in Harbor, and set ui_url_protocol/ssl_cert/ssl_cert
# Please set --with-trivy if needs enable Trivy in Harbor
# Please set --with-chartmuseum if needs enable Chartmuseum in Harbor
```

启动完成

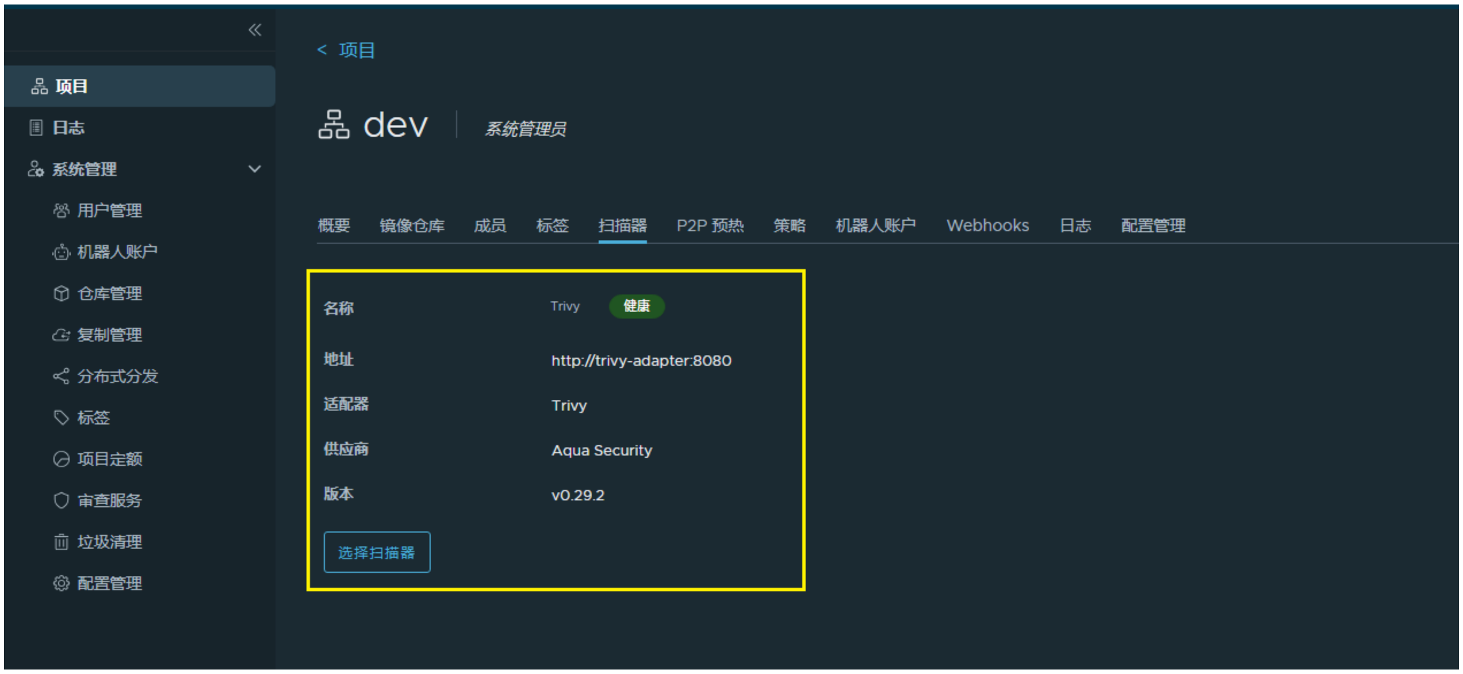
登录：http://172.22.70.12:8888/


Harbor
172.22.70.12:8888/account/sign-in?redirect_url=%2Fharbor%2Fprojects
Harbor
用户名
密码
记住我
登录
更多信息...
中文简体
关于

初始用户登录账号密码

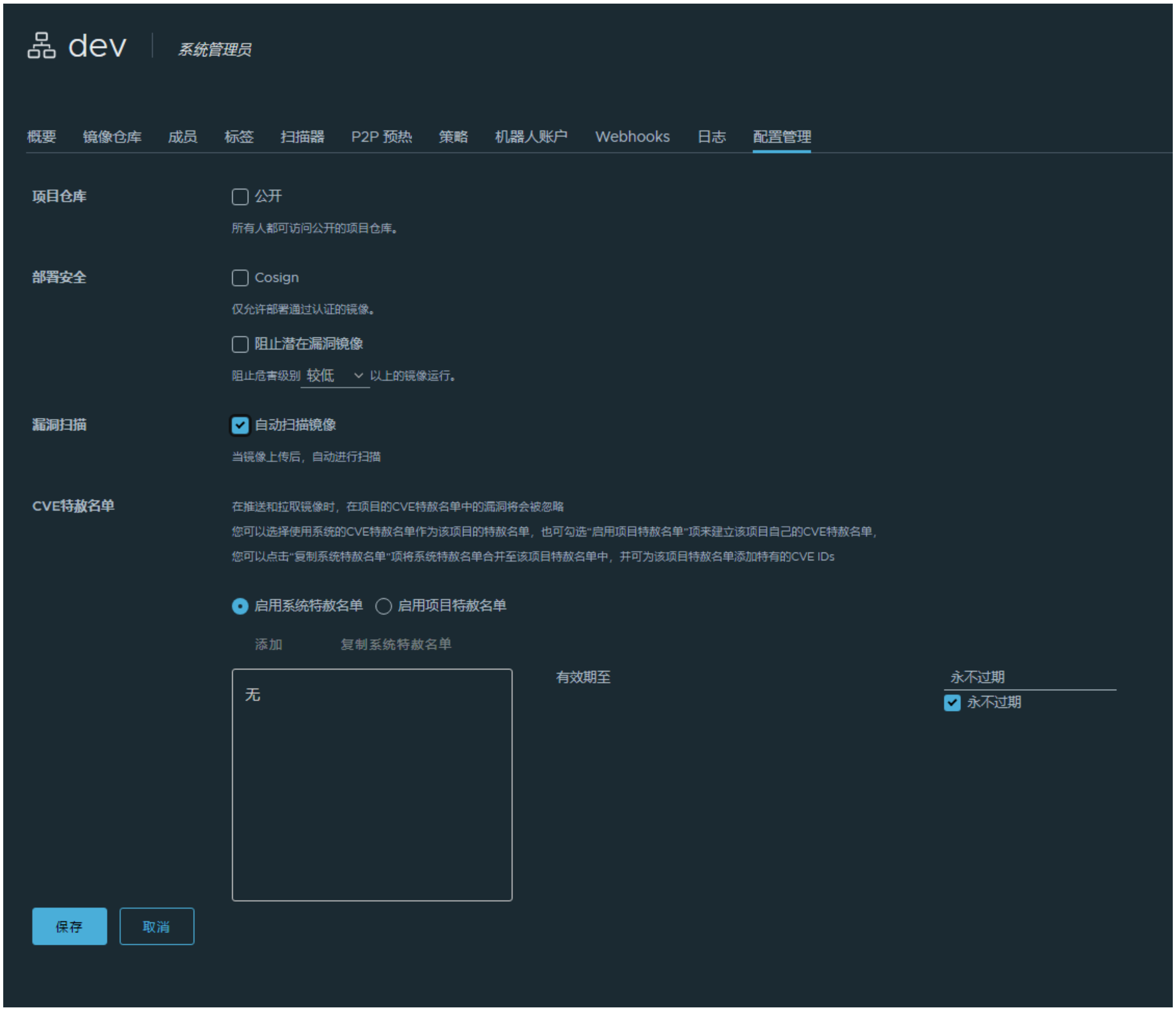
```
username : admin
password : Harbor12345
```

开启了镜像安全扫描器的启动，可在项目中看到trivy镜像扫描器



名称	Trivy 健康
地址	http://trivy-adapter:8080
适配器	Trivy
供应商	Aqua Security
版本	v0.29.2

可对镜像安全扫描进行配置


dev | 系统管理员

项目仓库 ☐ 公开
所有人都可访问公开的项目仓库。

部署安全 ☐ Cosign
仅允许部署通过认证的镜像。
☐ 阻止潜在漏洞镜像
阻止危害级别 较低 以上的镜像运行。

漏洞扫描 ☑ 自动扫描镜像
当镜像上传后，自动进行扫描

CVE特赦名单
在推送和拉取镜像时，在项目的CVE特赦名单中的漏洞将会被忽略
您可以选择使用系统的CVE特赦名单作为该项目的特赦名单，也可勾选"启用项目特赦名单"项来建立该项目自己的CVE特赦名单，
您可以点击"复制系统特赦名单"项将系统特赦名单合并至该项目特赦名单中，并可为该项目特赦名单添加特有的CVE IDs

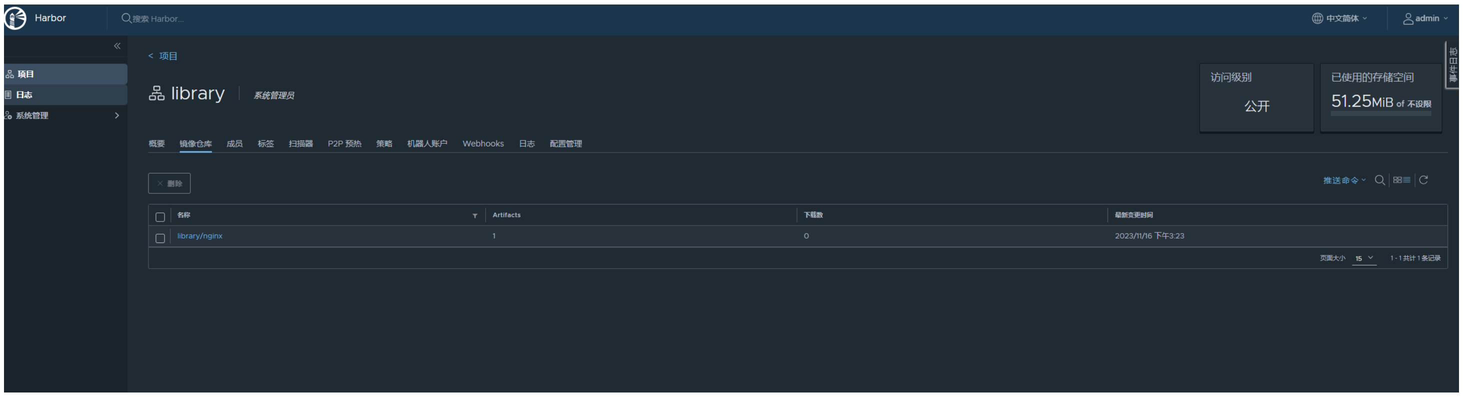
[✓ 启用系统特赦名单] 启用项目特赦名单

无

有效期至 永不过期
☑ 永不过期

镜像推送

```
# docker标记镜像
docker tag SOURCE_IMAGE[:TAG] 172.22.70.12:8888/library/REPOSITORY[:TAG]
# push 镜像到 harbor仓库
docker push 172.22.70.12:8888/library/REPOSITORY[:TAG]
# 示例如下:
# 给要推送的镜像打tag
docker tag nginx:1.19 172.22.70.12:8888/library/nginx:1.19
# 登录远程harbor仓库
docker login 172.22.70.12:8888
# 推送镜像到harbor仓库
docker push 172.22.70.12:8888/library/nginx:1.19
```

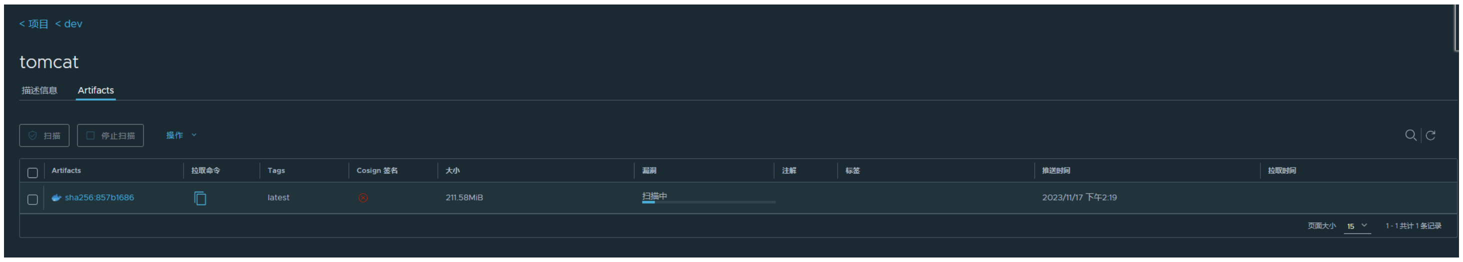


名称	Artifacts	下载数	最新变更时间
library/nginx	1	0	2023/11/16 下午3:23

页面大小 15　1 - 1 共计 1 条记录

自动对pushed的镜像进行漏洞扫描

- 前提：开启镜像安全扫描器

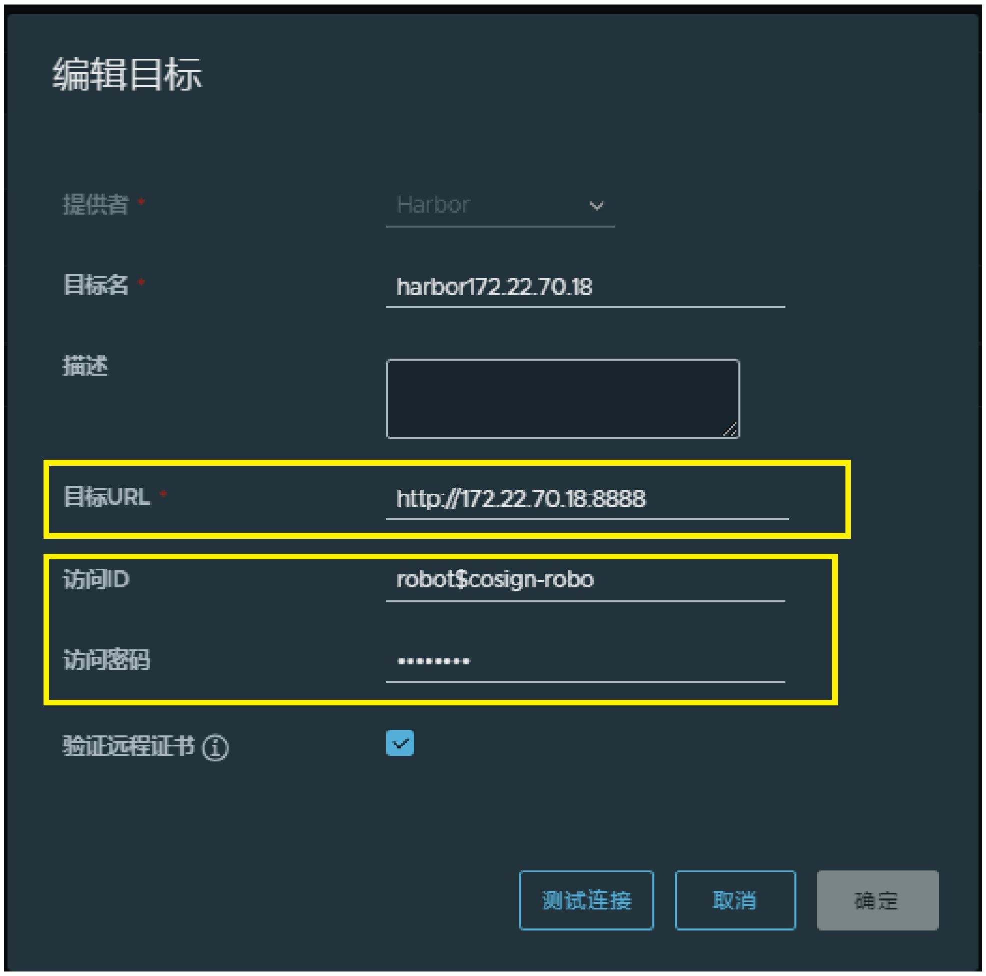


Artifacts	拉取命令	Tags	Cosign 签名	大小	漏洞	注解	标签	推送时间	拉取时间
sha256:857b1686		latest		211.58MiB	扫描中			2023/11/17 下午2:19	

页面大小 15　1 - 1 共计 1 条记录

仓库复制

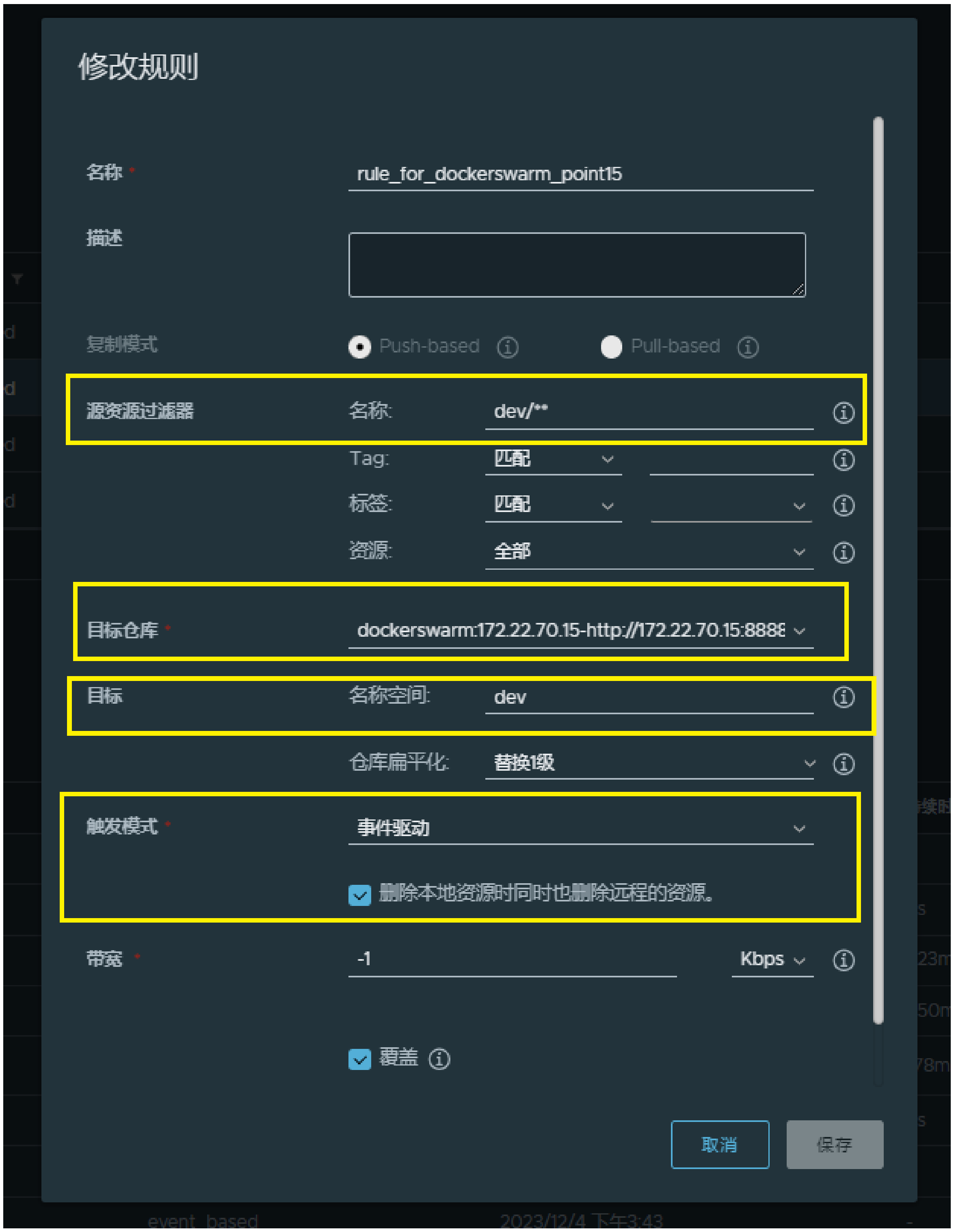
仓库管理 -> 添加目标 -> 目标仓库地址（ip + port）-> 访问ID（可用机器人账号访问/目标仓库指定用户账号）



编辑目标

提供者 *	Harbor
目标名 *	harbor172.22.70.18
描述	
目标URL *	http://172.22.70.18:8888
访问ID	robot$cosign-robo
访问密码	·········
验证远程证书 ⓘ	✓

复制管理界面->添加镜像复制规则 -> 资源过滤（dev/** : 项目dev下的所有资源） -> 目标仓库选择仓库管理配置的目标 -> 目标 （需要复制到目标仓库的项目名空间）-> 触发模式（什么机制下触发复制事件）



修改规则

名称 * rule_for_dockerswarm_point15

描述

复制模式 [✓ Push-based] Pull-based

源资源过滤器 名称: dev/**

Tag: 匹配

标签: 匹配

资源: 全部

目标仓库 * dockerswarm:172.22.70.15-http://172.22.70.15:8888

目标 名称空间: dev

仓库扁平化: 替换1级

触发模式 * 事件驱动

☑ 删除本地资源时同时也删除远程的资源。

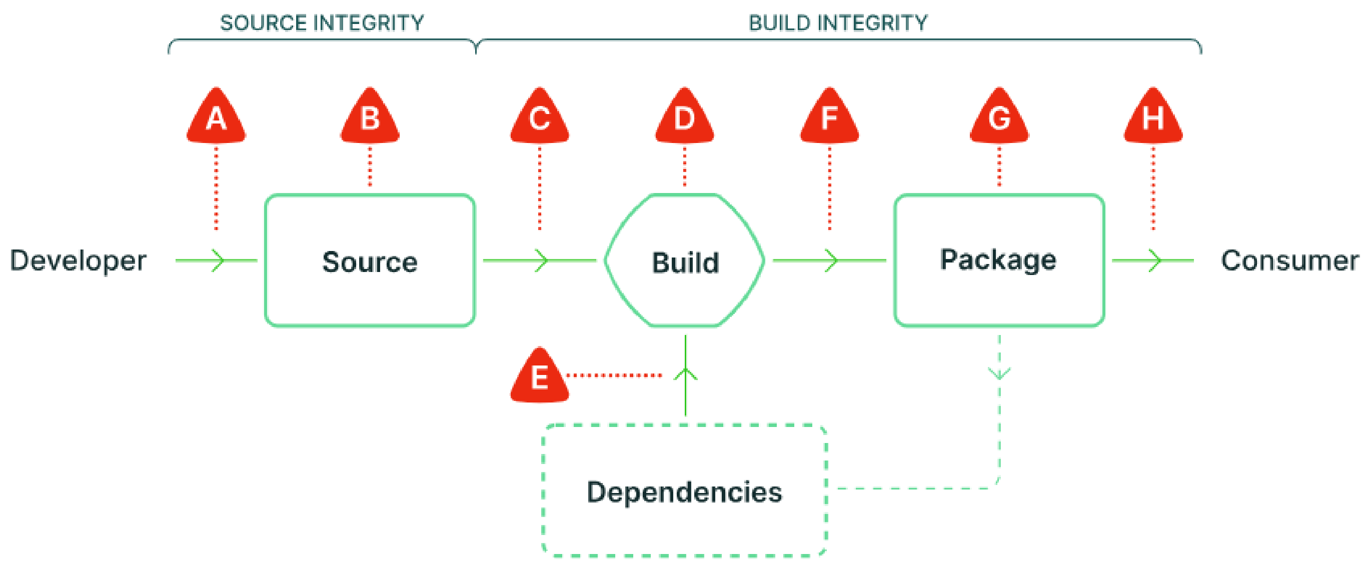
带宽 * -1 Kbps

☑ 覆盖

镜像签名(cosign)

签名过程

https://blog.gitguardian.com/supply-chain-security-sigstore-and-cosign-part-ii/#:~:text=Using Cosign 1 3.1 Install The easiest way,this%3A ... 3 3.3 Signature and Verification



A Submit unauthorized change
B Compromise source repo
C Build from modified source
D Compromise build process
E Use compromised dependency
F Upload modified package
G Compromise package repo
H Use compromised package

Cosign

https://goharbor.io/blog/cosign-2.5.0/

安装

https://edu.chainguard.dev/open-source/sigstore/cosign/how-to-install-cosign/

Installing Cosign with the Cosign Binary

```
# 下载安装包
wget "https://github.com/sigstore/cosign/releases/download/v2.0.0/cosign-linux-amd64"
# 移动到local/bin
sudo mv cosign-linux-amd64 /usr/local/bin/cosign
# 授予执行权限
sudo chmod +x /usr/local/bin/cosign
```

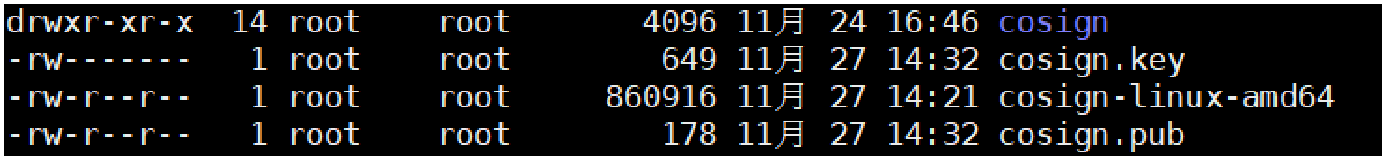
Cosign镜像签名

启动harbor仓库

```
# 进入harbor目录
cd ~/harbor
# 启动harbor
./install.sh --with-notary --with-trivy
# 也可以直接使用启动
./install.sh --with-trivy --with-chartmuseum
```

采用cosgin生成cosign.key

```
$ cosign generate-key-pair
>>> Enter password for private key:
>>> Enter again:
>>> Private key written to cosign.key
>>> Public key written to cosign.pub
```

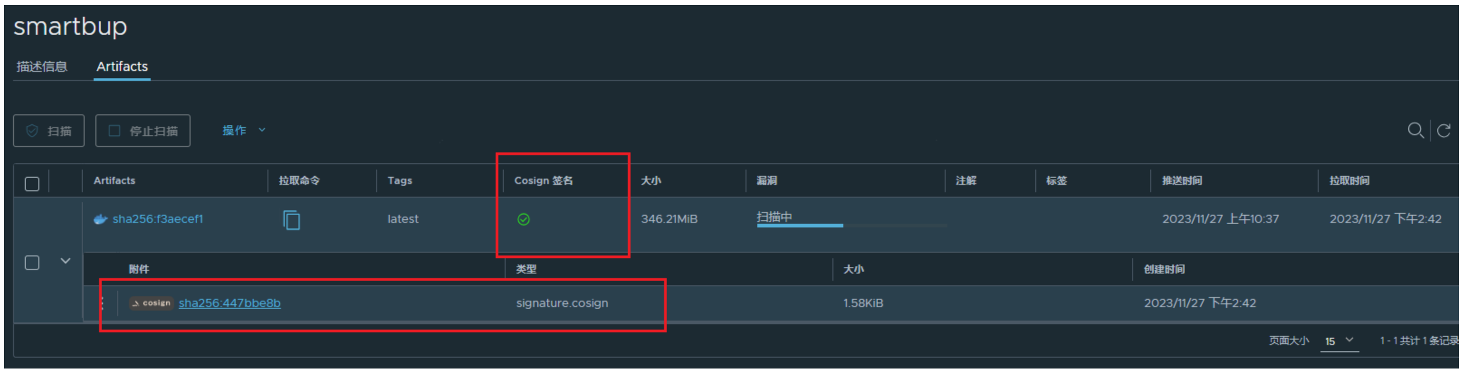


```
drwxr-xr-x  14 root    root       4096 11月  24 16:46 cosign
-rw-------   1 root    root        649 11月  27 14:32 cosign.key
-rw-r--r--   1 root    root     860916 11月  27 14:21 cosign-linux-amd64
-rw-r--r--   1 root    root        178 11月  27 14:32 cosign.pub
```

以项目授权用户登录，并push打好标签的镜像，使用cosign进行镜像签名，sign的密码为之前生成cosign.key输入的密码

```
# 登录harbor仓库
$ docker login <ip>:<port>
# push镜像
 docker push <镜像tag>
 # cosign镜像
 cosign sign --key cosign.key <镜像tag>
>>> Enter password for private key:
>>> Pushing signature to: xxxx
```

可见镜像添加了signature.cosign签名附件



Artifacts	拉取命令	Tags	Cosign 签名	大小	漏洞	注解	标签	推送时间	拉取时间
sha256:f3aecef1		latest	✓	346.21MiB	扫描中			2023/11/27 上午10:37	2023/11/27 下午2:42

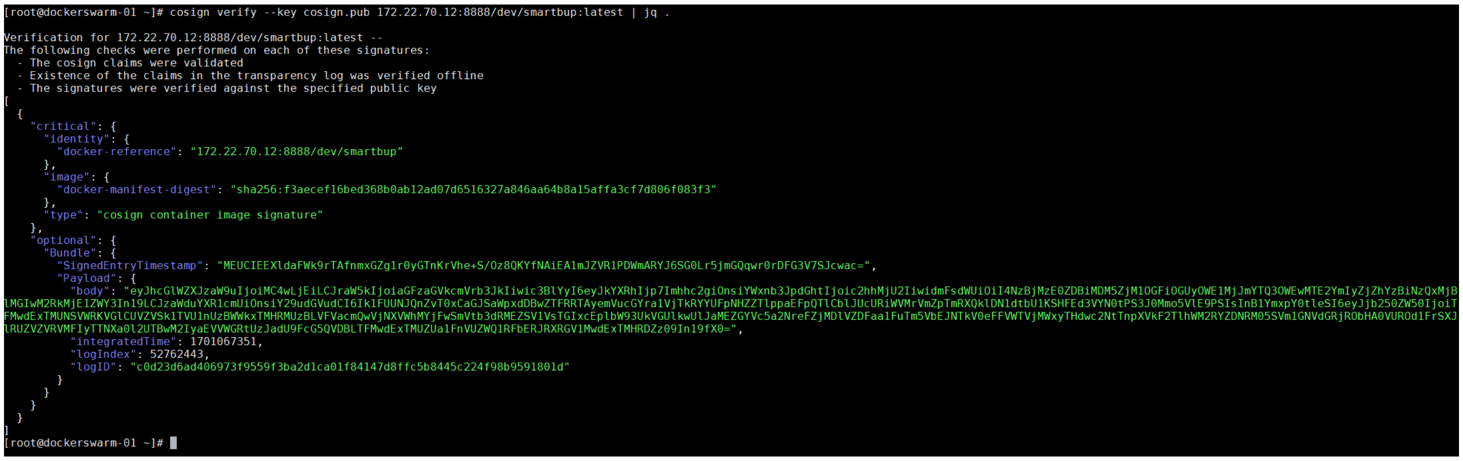
附件	类型	大小	创建时间
cosign sha256:447bbe8b	signature.cosign	1.58KiB	2023/11/27 下午2:42

Cosign签名验证

验证签名镜像

```
cosign verify --key cosign.pub <镜像tag> | jq .
# 如果提示没有 jq 命令 json工具包 : 在centos下使用:
yum -y install jq
```

验证结果



```
[root@dockerswarm-01 ~]# cosign verify --key cosign.pub 172.22.70.12:8888/dev/smartbup:latest | jq .

Verification for 172.22.70.12:8888/dev/smartbup:latest --
The following checks were performed on each of these signatures:
  - The cosign claims were validated
  - Existence of the claims in the transparency log was verified offline
  - The signatures were verified against the specified public key
[
  {
    "critical": {
      "identity": {
        "docker-reference": "172.22.70.12:8888/dev/smartbup"
      },
      "image": {
        "docker-manifest-digest": "sha256:f3aecef16bed368b0ab12ad07d6516327a846aa64b8a15affa3cf7d806f083f3"
      },
      "type": "cosign container image signature"
    },
    "optional": {
      "Bundle": {
        "SignedEntryTimestamp": "MEUCIEEXldaFWk9rTAfnmxGZg1r0yGTnKrVhe+S/Oz8QKYfNAiEA1mJZVR1PDWmARYJ6SG0Lr5jmGQqwr0rDFG3V7SJcwac=",
        "Payload": {
          "body": "eyJhcGlWZXJzaW9uIjoiMC4wLjEiLCJraW5kIjoiaGFzaGVkcmVrb3JkIiwic3BlYyI6eyJkYXRhIjp7Imhhc2giOnsiYWxnb3JpdGhtIjoic2hhMjU2IiwidmFsdWUiOiI4NzBjMzE0ZDBiMDM5ZjM1OGFiOGUyOWE1MjJmYTQ3OWEwMTE2YmIyZjZhYzBiNzQxMjB
lMGIwM2RkMjE1ZWY3In19LCJzaWduYXR1cmUiOnsiY29udGVudCI6Ik1FUUNJQnZvT0xCaGJSaWpxdDBwZTFRRTAyemVucGYra1VjTkRYYUFpNHZZTlppaEFpQTlCblJUcURiWVMrVmZpTmRXQklDN1dtbU1KSHFEd3VYN0tPS3J0Mmo5VlE9PSIsInB1YmxpY0tleSI6eyJjb250ZW50IjoiT
FMwdExTMUNSVWRKVGlCUVZVSk1TVU1nUzBWWkxTMHRMUzBLVFVacmQwVjNXVWhMYjFwSmVtb3dRMEZSV1VsTGIxcEplbW93UkVGUlkwUlJaMEZGYVc5a2NreFZjMDlVZDFaa1FuTm5VbEJNTkV0eFFVWTVjMWxyTHdwc2NtTnpXVkF2TlhWM2RYZDNRM05SVm1GNVdGRjRObHA0VUROd1FrSXJ
lRUZVZVRVMFIyTTNXa0l2UTBwM2IyaEVVWGRtUzJadU9FcG5QVDBLTFMwdExTMUZUa1FnVUZWQ1RFbERJRXRGV1MwdExTMHRDZz09In19fX0=",
          "integratedTime": 1701067351,
          "logIndex": 52762443,
          "logID": "c0d23d6ad406973f9559f3ba2d1ca01f84147d8ffc5b8445c224f98b9591801d"
        }
      }
    }
  }
]
[root@dockerswarm-01 ~]#
```

镜像制品保留策略

参考文档:

https://goharbor.io/docs/1.10/working-with-projects/working-with-images/create-tag-retention-rules/

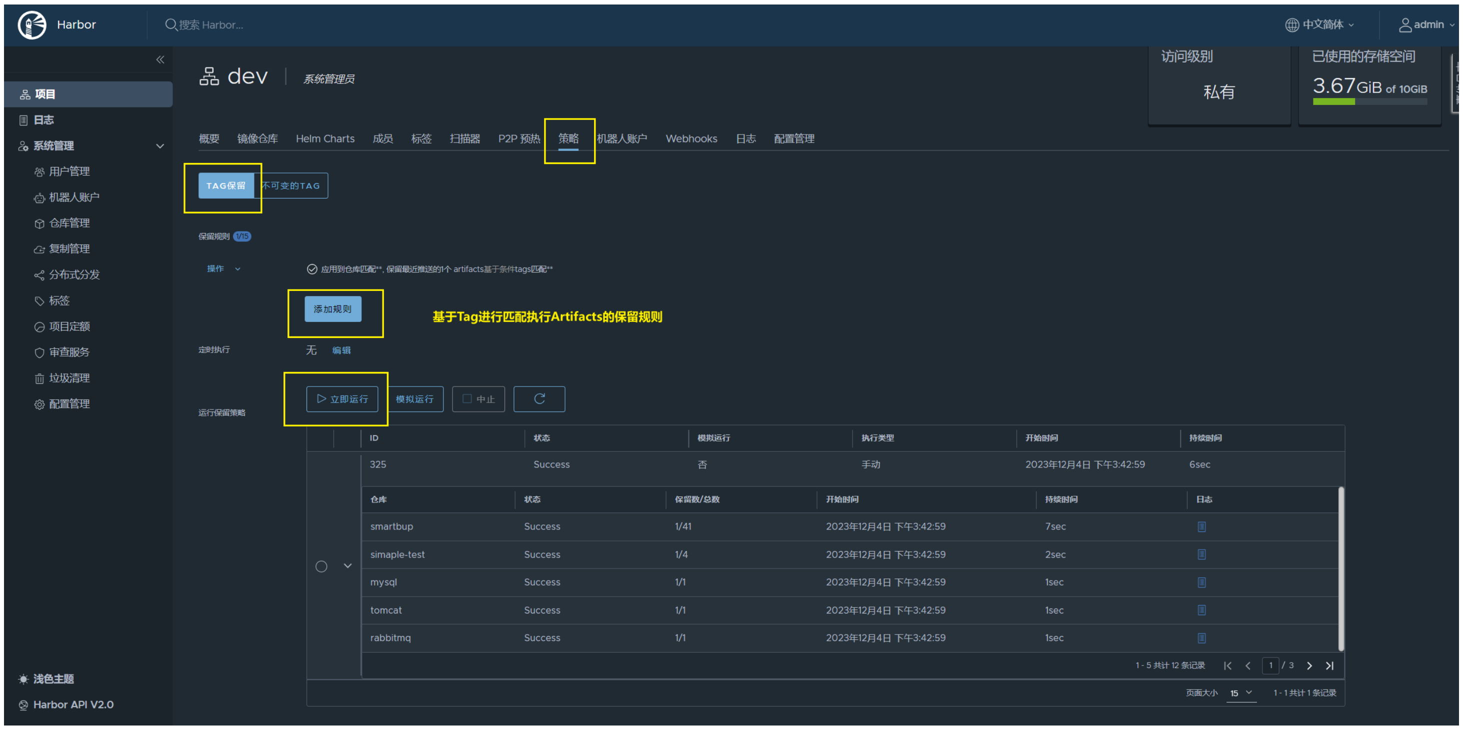
https://blog.csdn.net/q48S71bCzBeYLOu9T0n/article/details/117202971

Artifact保留策略的设置是以项目为单位的，并且以 Tag 作为 Artifact 的标识来判断是否需要保留，所以管理界面上显示的是“Tag保留规则”

该规则是一个包含仓库名称匹配、Artifact 条件和Tag 名称匹配的过滤器。Harbor 保留策略任务在执行过程中对每个 Artifact 都用保留规则匹配，如果Artifact 被任意一条规则匹配成功，即为需要保留的 Artifact，否则为待删除的 Artifact。

保留制品的策略，以镜像仓库项目为单位，可手动单次进行保留策略的执行，也可按指定规则自动执行镜像制品配置的保留策略。

仓库项目 -> 策略 -> 添加规则（tag保留）-> 立即执行 / 定时执行



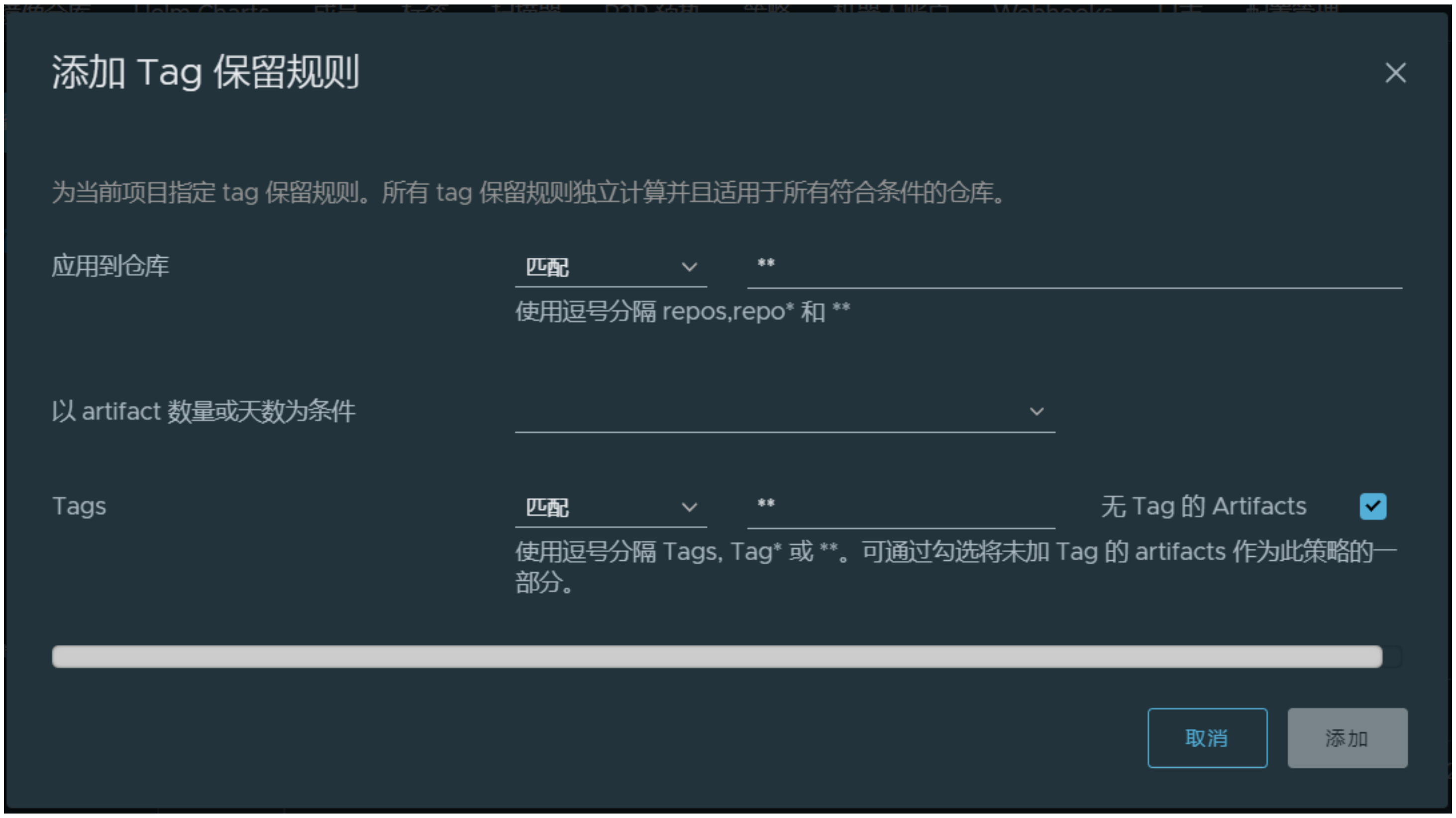
应用到仓库匹配**, 保留最近推送的1个 artifacts基于条件tags匹配**

ID	状态	模拟运行	执行类型	开始时间	持续时间
325	Success	否	手动	2023年12月4日 下午3:42:59	6sec

仓库	状态	保留数/总数	开始时间	持续时间	日志
smartbup	Success	1/41	2023年12月4日 下午3:42:59	7sec	
simaple-test	Success	1/4	2023年12月4日 下午3:42:59	2sec	
mysql	Success	1/1	2023年12月4日 下午3:42:59	1sec	
tomcat	Success	1/1	2023年12月4日 下午3:42:59	1sec	
rabbitmq	Success	1/1	2023年12月4日 下午3:42:59	1sec	

1 - 5 共计 12 条记录 1 / 3

页面大小 15 1 - 1 共计 1 条记录



为当前项目指定 tag 保留规则。所有 tag 保留规则独立计算并且适用于所有符合条件的仓库。

应用到仓库: 匹配 **

使用逗号分隔 repos,repo* 和 **

以 artifact 数量或天数为条件

Tags: 匹配 ** 无 Tag 的 Artifacts ✓

使用逗号分隔 Tags, Tag* 或 **。可通过勾选将未加 Tag 的 artifacts 作为此策略的一部分。

问题解决

[Step 1]: checking docker-compose is installed ...✕ Need to install docker-compose(1.18.0+) by yourself first and run this script again.

https://blog.csdn.net/webBOFB/article/details/129330055 docker新版语法问题

- 修改common.sh文件

```
vim common.sh
```

```
将改所有docker-compose --version 为 docker compose version
将所有包含docker-compose命令改为docker compose
```

- 修改install.sh

```
vim ./install.sh
```

```
将所有包含docker-compose命令改为docker compose
```

报错提示：Error response from daemon: Get https://192.168.186.120/v1/users/: dial tcp 192.168.186.120:443: getsockopt: connection refused

参考文档： https://www.cnblogs.com/hahaha111122222/p/11799300.html

修改docker的daemon.json文件（如果没有就在/etc/docker/文件夹下新建daemon.json文件）
将需要访问的地址添加到"insecure-registries": ["https://172.22.70.12:8888"]

```
vim /etc/docker/daemon.json
# 在daemon.json文件里添加如下代码
{
   "insecure-registries": ["https://172.22.70.12:8888"]
}
```

重载daemon文件 并 重启docker

```
systemctl daemon-reload
systemctl restart docker
```