# 《云计算专业实训 III》实验报告\_01

## 一、基本信息

* 课程名称：《云计算专业实训 III》
* 实验名称：基于 HTTPS 协议搭建 Harbor 仓库及相关操作实验
* 姓名：赵文宇
* 班级：云计算2班
* 学号：230303060
* 实验日期：2025/9/15
* 成绩：A+

## 二、实验目的

基于 HTTPS 协议搭建 Harbor 仓库，在仓库中创建 test 项目，将该项目公开，并将 docker 本地的 busybox:latest 镜像上传至 Harbor 的 test 项目中，使用远程 docker 主机master下载 Harbor 中的 busybox:latest 镜像，并将该镜像运行为容器 busybox-1。

## 三、实验环境

1. 硬件环境

* 本地主机：

Intel Xeon Platinum 8380（28核56线程）/ AMD EPYC 7763（64核128线程）

256GB DDR4 ECC

2TB NVMe SSD（系统盘） + 4TB SATA SSD（数据盘）

双万兆网卡（有线） + Wi-Fi 6E软件环境

* 远程 Docker 主机：

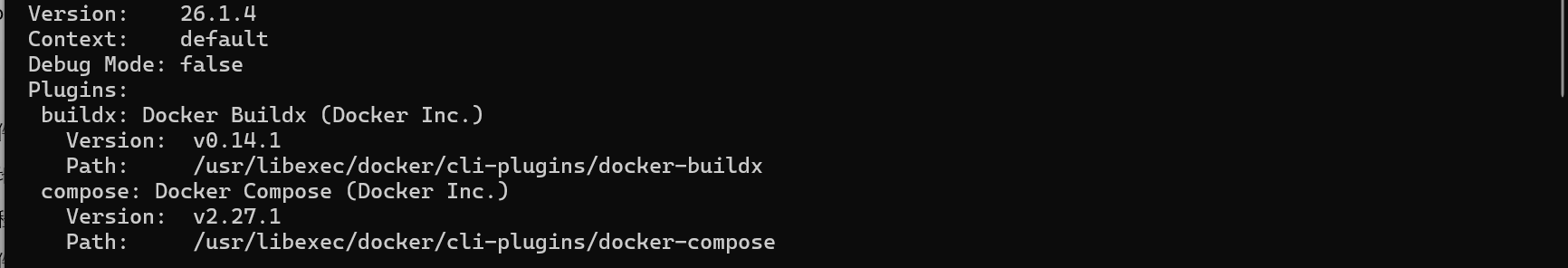
2 × Intel Xeon Platinum 8380（56核112线程）/ 2 × AMD EPYC 7763（128核256线程）

512GB DDR4 ECC

4TB NVMe SSD（镜像存储） + 8TB SATA SSD（备份） + 24TB HDD（冷备份）

10Gbps 光纤网卡 + 40Gbps Infiniband

* 操作系统 CentOS 7
* Docker 版本：Docker version 26.1.4, build 5650f9b
* Harbor 版本：2.3.0
* 其他相关软件：VM



## 四、实验原理

1. HTTPS 协议：简述 HTTPS 协议的工作原理，包括 SSL/TLS 加密、证书验证等内容。
2. Harbor 仓库：介绍 Harbor 仓库的功能、架构及在 Docker 镜像管理中的作用。
3. Docker 相关：说明 Docker 镜像的上传、下载以及容器运行的基本原理。

**HTTPS协议**：  
**SSL/TLS加密数据，证书验证身份防窃听篡改**

**Harbor仓库**：  
**集中管理镜像，支持权限控制与漏洞扫描**

**Docker相关**：  
**镜像拉取推送，容器隔离运行进程与资源**

## 五、重要操作步骤（含截图）

1. 搭建基于 HTTPS 协议的 Harbor 仓库

* 步骤 1：准备 SSL 证书。[详细描述生成或获取 SSL 证书的过程，如使用 OpenSSL 生成自签名证书等]，截图如下：

**生成 ca 证书**

openssl genrsa -out ca.key 3072

#生成一个 3072 位的 key，也就是私钥

openssl req -new -x509 -days 3650 -key ca.key -out ca.pem

#生成一个数字证书 ca.pem，3650 表示证书的有效时间是 3 年，按箭头提示填写即可，没有箭头标注的为

空：



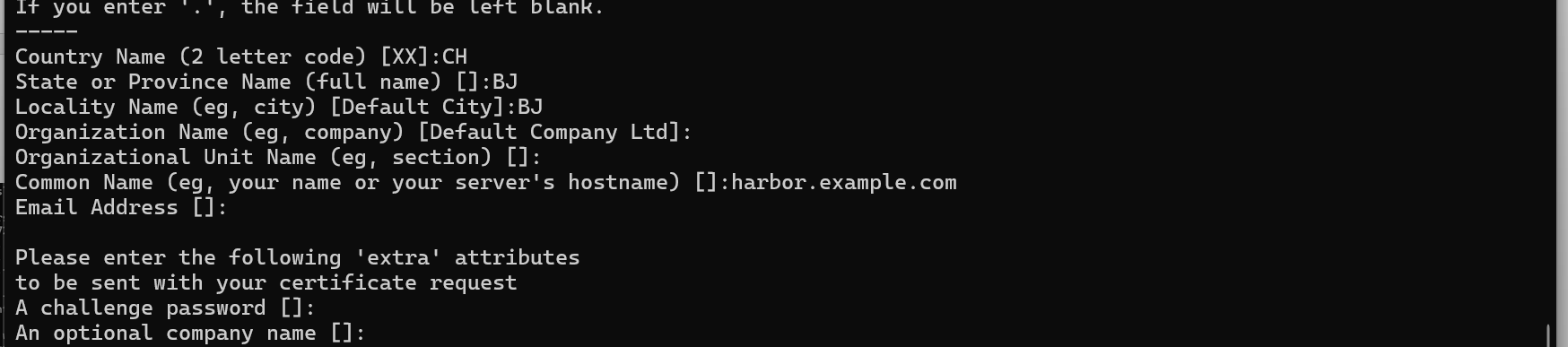
**生成域名的证书：**

openssl genrsa -out harbor.key 3072

#生成一个 3072 位的 key，也就是私钥

openssl req -new -key harbor.key -out harbor.csr

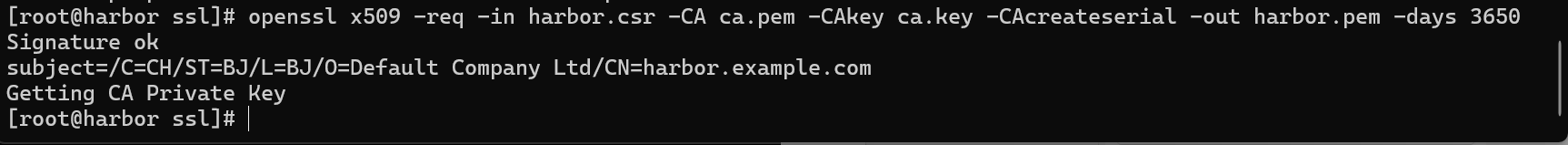
#生成一个证书请求，一会签发证书时需要的，标箭头的按提示填写，没有箭头标注的为空：



**签发证书：**

openssl x509 -req -in harbor.csr -CA ca.pem -CAkey ca.key -CAcreateserial -out harbor.pem - days 3650

显示如下，说明证书签发好了：



* 步骤 2：配置 Harbor。[说明修改 Harbor 配置文件（如 harbor.yml）中关于 HTTPS 的相关设置，包括证书路径等]，截图如下：[此处插入步骤 2 的截图]

**Harbor**

**Docker**

#配置时间同步

[root@harbor~]# yum install -y ntp ntpdate

[root@master1 ~]# ntpdate cn.pool.ntp.org

#编写计划任务

[root@harbor~]# crontab -e

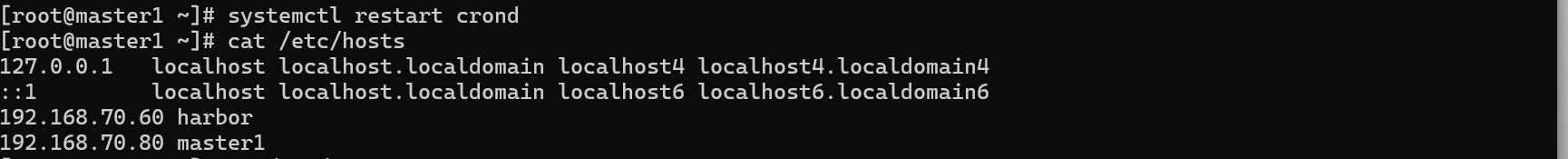
\* \*/1 \* \* \* /usr/sbin/ntpdate cn.pool.ntp.org

重启 crond 服务使配置生效：

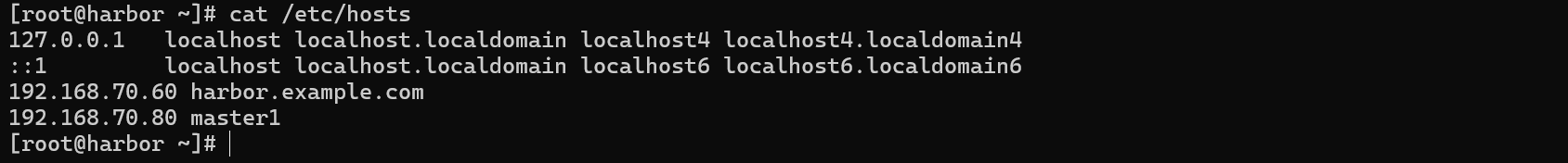
[root@master1 ~]# systemctl restart crond

配置 hosts 文件

[root@master1 ~]# cat /etc/hosts



[root@harbor harbor]# cat /etc/hosts

安装基础软件包

[root@ harbor~]# yum install -y wget net-tools nfs-utils lrzsz gcc gcc-c++ make cmake libxml2-devel openssl-devel curl curl-devel unzip sudo ntp libaio-devel wget vim ncursesdevel autoconf automake zlib-devel python-devel epel-release openssh-server socat ipvsadm conntrack

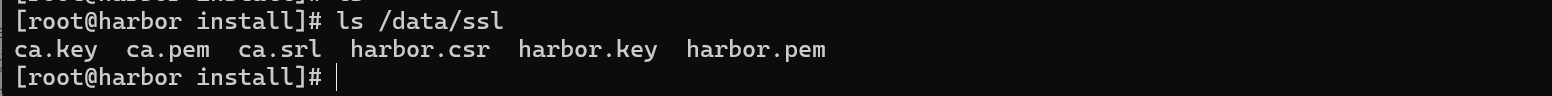
创建安装目录

[root@harbor ssl]# mkdir /data/install -p

[root@harbor ssl]# cd /data/install/

安装 harbor

/data/ssl 目录下有如下文件：

 [root@harbor install]# cd /data/install/

#把 harbor 的离线包 harbor-offline-installer-v2.3.0-rc3.tgz 上传到这个目录，离线包在课件

里提供了

下载 harbor 离线包的地址：

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

解压：

[root@harbor install]# tar zxvf harbor-offline-installer-v2.3.0-rc3.tgz

[root@harbor install]# cd harbor

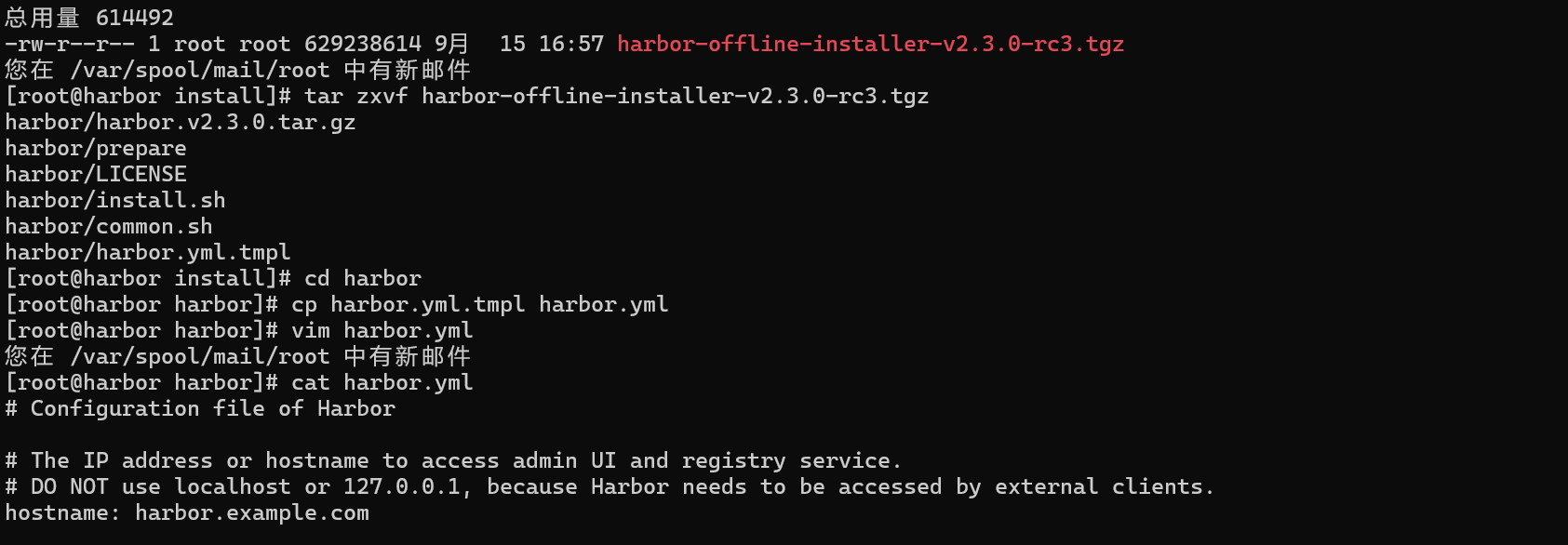
[root@harbor harbor]# cp harbor.yml.tmpl harbor.yml

[root@harbor harbor]# vim harbor.yml

修改配置文件：

hostname: harbor

#修改 hostname，跟上面签发的证书域名保持一致



#协议用 https

certificate: /data/ssl/harbor.pem

private\_key: /data/ssl/harbor.key



邮件和 ldap 不需要配置，在 harbor 的 web 界面可以配置

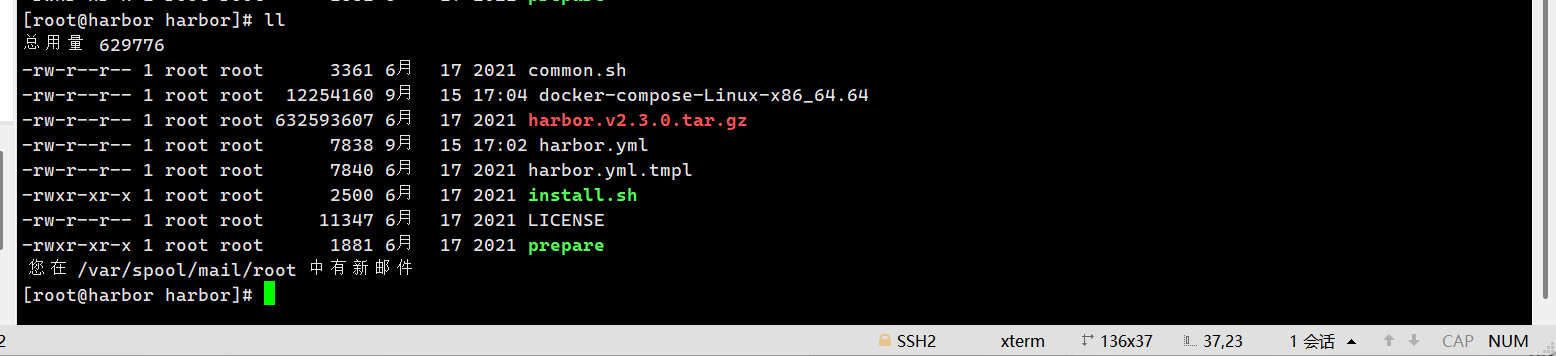
其他配置采用默认即可

修改之后保存退出

注：harbor 默认的账号密码：admin/Harbor12345

安装 docker-compose

上传课件里的 docker-compose-Linux-x86\_64 文件到 harbor 机器



[root@harbor harbor]# mv docker-compose-Linux-x86\_64.64 /usr/bin/docker-compose

[root@harbor harbor]# chmod +x /usr/bin/docker-compose

注： docker-compose 项目是 Docker 官方的开源项目，负责实现对 Docker 容器集群的快速编排。

Docker-Compose 的工程配置文件默认为 docker-compose.yml，Docker-Compose 运行目录下的必要有一个

docker-compose.yml。docker-compose 可以管理多个 docker 实例。

安装 harbor 需要的离线镜像包 docker-harbor-2-3-0.tar.gz 在课件，可上传到 harbor 机器，通过

docker load -i 解压

[root@harbor install]# docker load -i docker-harbor-2-3-0.tar.gz

[root@harbor install]# cd /data/install/harbor

[root@harbor harbor]# ./install.sh

看到下面内容，说明安装成功：

[Step 5]: starting Harbor ...

Creating network "harbor\_harbor" with the default driver

Creating harbor-log ... done

Creating registryctl ... done

Creating harbor-db ... done

Creating redis ... done

Creating registry ... done

Creating harbor-portal ... done

Creating harbor-core ... done

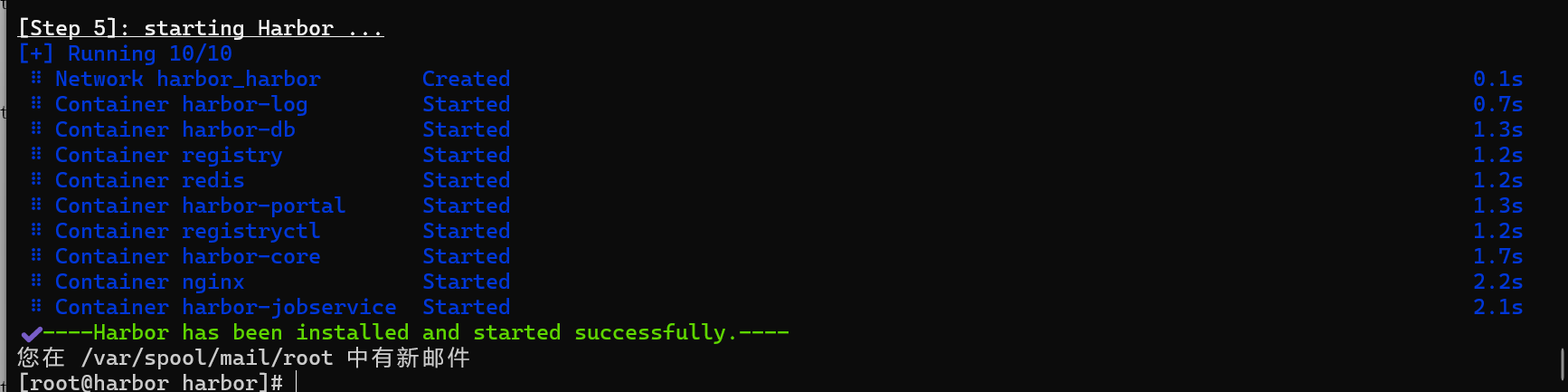
Creating harbor-jobservice ... done

Creating nginx ... done

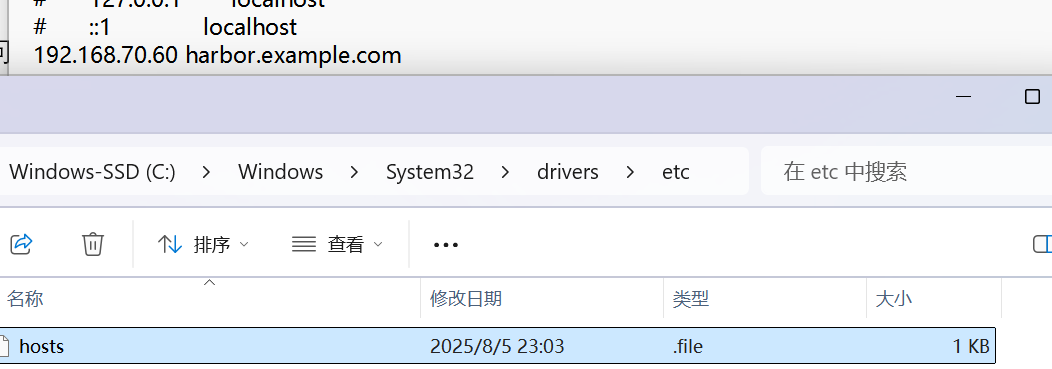
----Harbor has been installed and started successfully.----

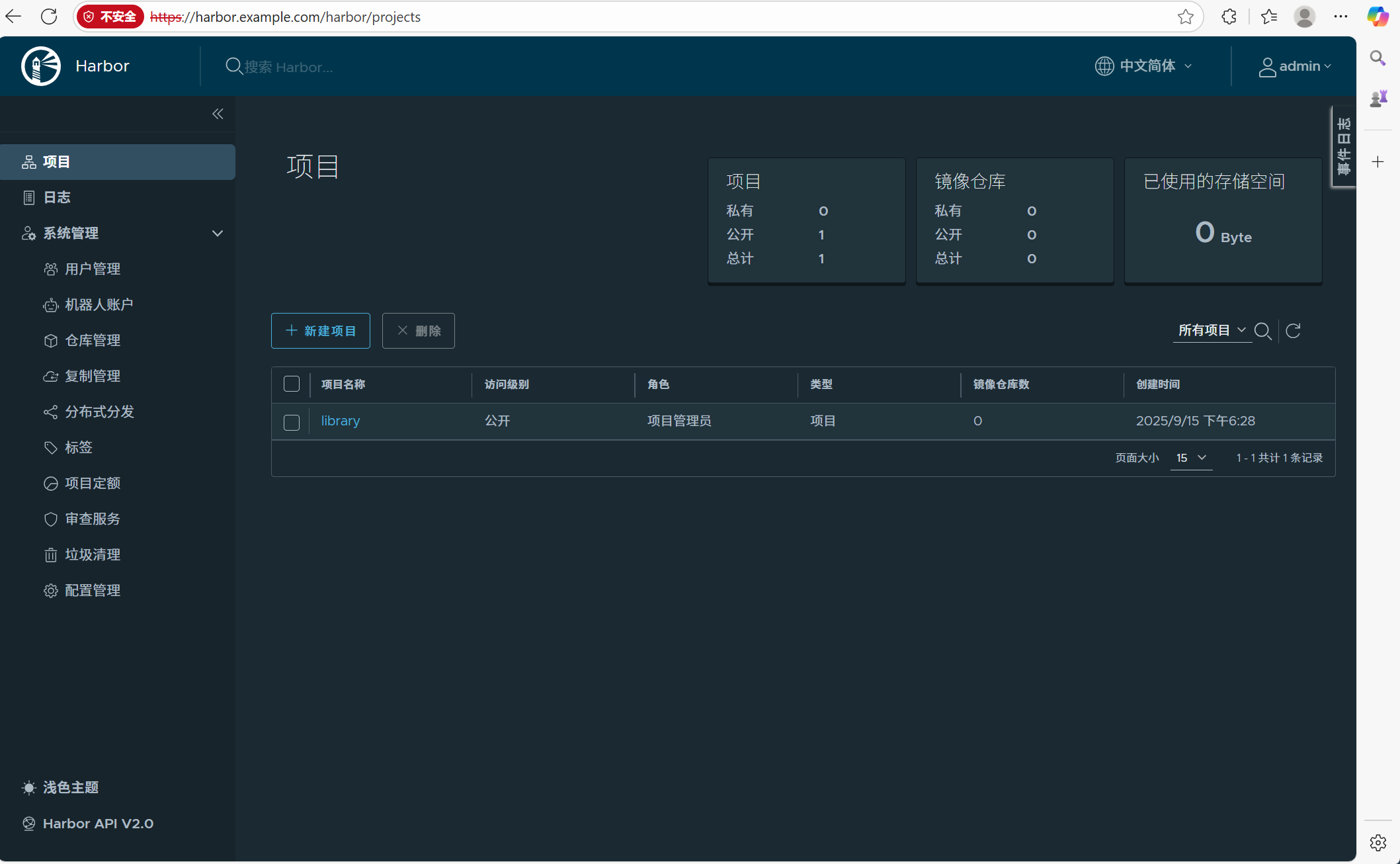
在自己电脑修改 hosts 文件

* 步骤 3：启动 Harbor。[执行启动 Harbor 的命令，如./install.sh]，截图如下：[此处插入步骤 3 的截图，包含启动成功的提示信息]



* 步骤 4：验证 Harbor 是否成功搭建。[通过浏览器访问 Harbor 的 HTTPS 地址，查看是否能正常打开登录页面]，截图如下：[此处插入步骤 4 的截图]





1. 在 Harbor 仓库中创建 test 项目并公开

步骤 1：登录 Harbor。[使用管理员账号登录 Harbor 界面]，截图如下：[此处插入步骤 1 的截图]

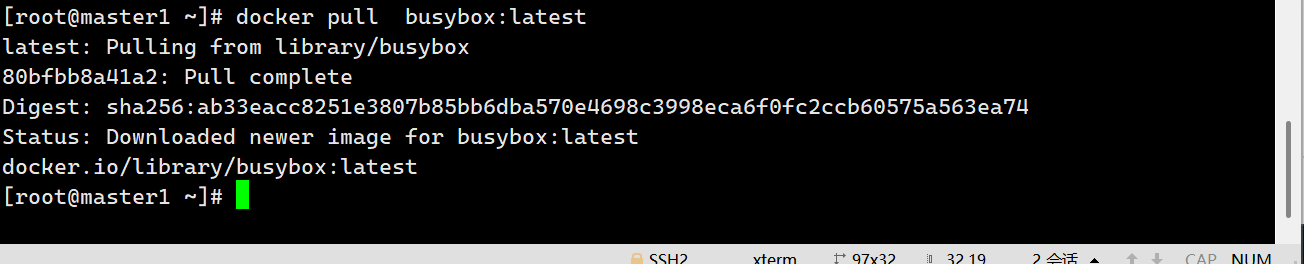
步骤 2：创建 test 项目。[在 Harbor 界面中找到创建项目的入口，输入项目名称 test 等信息]，截图如下：[此处插入步骤 2 的截图]

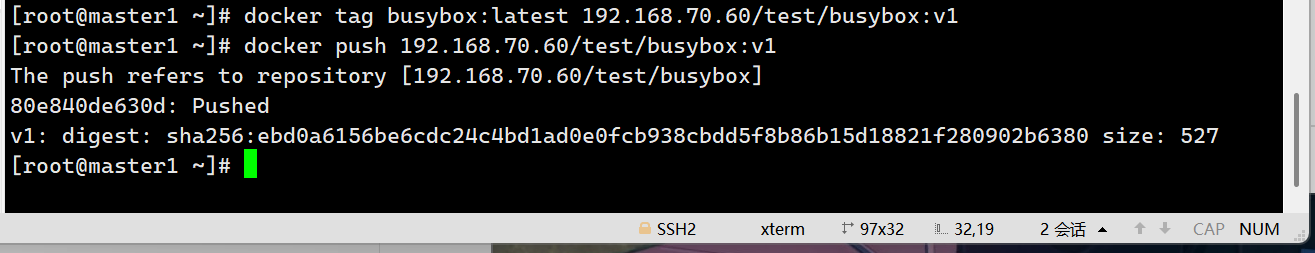
步骤 3：将 test 项目公开。[在 test 项目的设置中，将项目设置为公开状态]，截图如下：[此处插入步骤 3 的截图，显示项目已公开]



1. 将本地 docker 的 busybox:latest 镜像上传至 Harbor 的 test 项目

* 步骤 1：登录 Harbor 仓库（本地 Docker 命令行）。[执行 docker login 命令登录到搭建的 Harbor 仓库，输入用户名和密码]，截图如下：[此处插入步骤 1 的截图，包含登录成功信息]
* 步骤 2：为 busybox:latest 镜像打标签。[执行 docker tag 命令，将镜像标记为 Harbor 中 test 项目的路径格式，如 [Harbor 地址]/test/busybox:latest]，截图如下：[此处插入步骤 2 的截图]
* 步骤 3：上传镜像至 Harbor 的 test 项目。[执行 docker push 命令上传标记后的镜像]，截图如下：[此处插入步骤 3 的截图，包含上传成功信息]

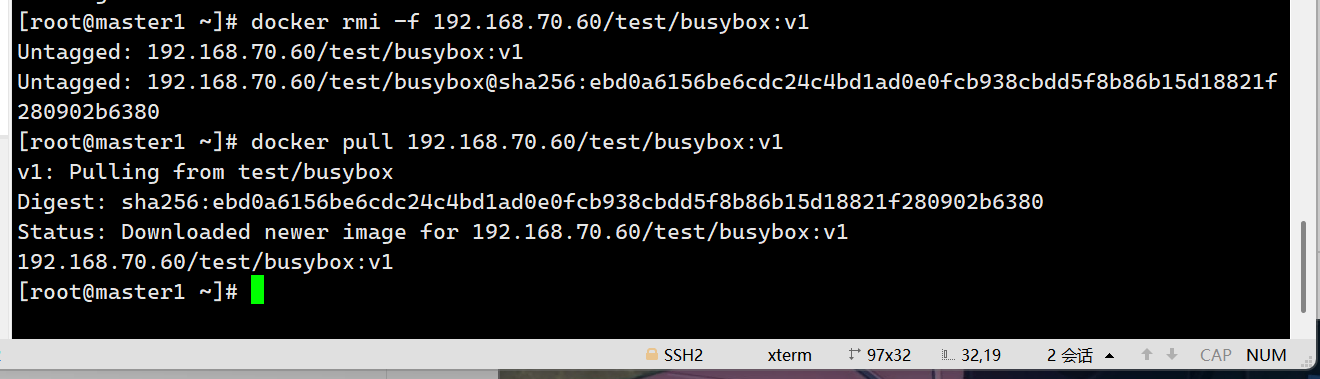






1. 使用远程 docker 主机下载 Harbor 中的 busybox:latest 镜像并运行为容器 busybox-1

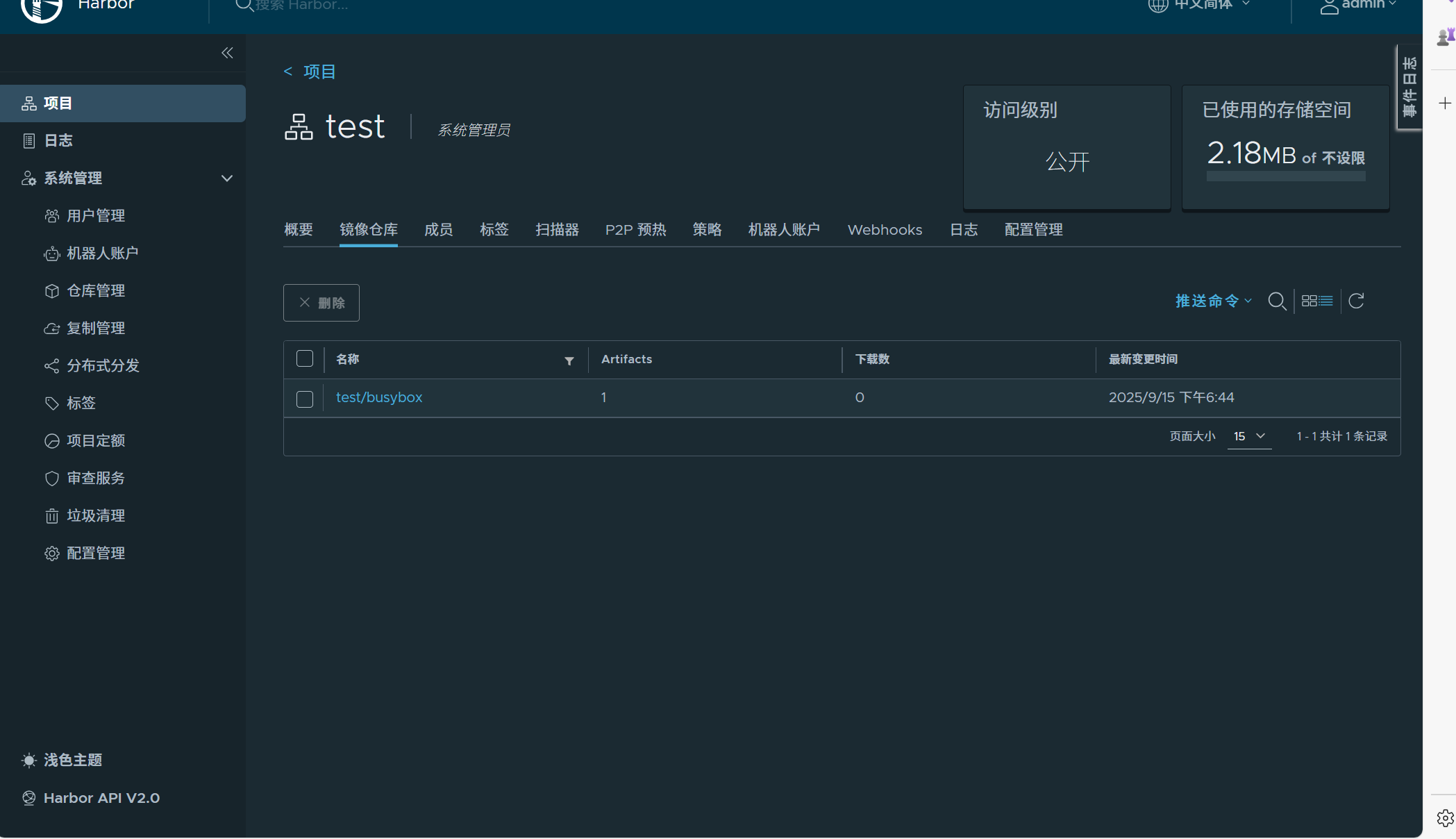
* 步骤 1：远程 Docker 主机master上登录 Harbor 仓库。[在远程 Docker 主机的命令行中执行 docker login 命令登录到 Harbor 仓库]，截图如下：[此处插入步骤 1 的截图，包含登录成功信息]
* 步骤 2：下载 Harbor 中 test 项目的 busybox:latest 镜像。[执行 docker pull 命令，如 docker pull [Harbor 地址]/test/busybox:latest]，截图如下：[此处插入步骤 2 的截图，包含下载成功信息]
* 步骤 3：将下载的镜像运行为容器 busybox-1。[执行 docker run 命令，如 docker run --name busybox-1 -d [Harbor 地址]/test/busybox:latest]，截图如下：[此处插入步骤 3 的截图，包含容器创建成功信息]
* 步骤 4：验证容器运行状态。[执行 docker ps 命令查看容器 busybox-1 的运行状态]，截图如下：[此处插入步骤 4 的截图，显示容器正常运行]



## 六、实验结果

1. Harbor 仓库搭建结果：成功基于 HTTPS 协议搭建 Harbor 仓库，通过浏览器可正常访问，[补充其他相关验证结果]。



1. test 项目创建及公开结果：成功在 Harbor 中创建 test 项目，并将其设置为公开状态，[补充相关查看结果]。
2. 镜像上传结果：本地 docker 的 busybox:latest 镜像成功上传至 Harbor 的 test 项目，在 Harbor 界面的 test 项目中可查看该镜像，[补充相关截图或描述]。
3. 

镜像下载及容器运行结果：远程 docker 主机成功下载 Harbor 中 test 项目的 busybox:latest 镜像，且成功将其运行为容器 busybox-1，容器处于正常运行状态，[补充相关验证结果]。

## 七、实验总结与思考

1. 实验总结：总结本次实验完成的主要内容、达成的实验目的以及在实验过程中的主要收获。
2. 遇到的问题及解决方法：记录实验过程中遇到的问题，以及解决这些问题所采取的方法和思路。
3. 实验思考：对实验原理、操作步骤等进行思考，分析实验中可能存在的优化点或改进方向，以及实验所涉及技术在实际应用中的场景等。

**1. 实验总结**

**主要内容**：  
本次实验完成了以下核心任务：

1. 将本地 busybox:latest 镜像上传至 Harbor 私有仓库的 test 项目。
2. 通过远程 Docker 主机（master1）从 Harbor 下载镜像并运行容器 busybox-1。
3. 验证了镜像推送、拉取及容器运行的完整流程。

**主要收获**：

* 理解了 Docker 镜像仓库在分布式环境中的核心作用。
* 掌握了 Harbor 作为私有仓库的配置与安全认证机制（如非 HTTPS 访问的配置）。
* 通过实际操作巩固了 Docker 命令行工具的使用（login/pull/run 等）。

**问题1：推送镜像时提示 An image does not exist locally**

* **原因**：未正确打标签，本地镜像名称与 Harbor 路径不匹配。
* **解决**：

bash

|  |
| --- |
| docker tag busybox:latest 192.168.70.60/test/busybox:latest # 重新打标签 |

* **思路**：镜像推送必须基于“仓库地址/项目/镜像名:标签”的完整格式。

**问题2：远程主机拉取镜像失败（Error response from daemon）**

* **原因**：Harbor 未配置为信任仓库（默认要求 HTTPS）。
* **解决**：
  1. 修改远程主机的 Docker 配置：

json

|  |
| --- |
| # /etc/docker/daemon.json |
| { |
| "insecure-registries": ["192.168.70.60"] |
| } |

* **思路**：私有仓库在测试环境中常使用 HTTP，需显式声明为不安全仓库。

**问题3：容器运行后立即退出**

* **原因**：默认 docker run 执行完命令后退出，未保持后台进程。
* **解决**：

bash

|  |
| --- |
| docker run --name busybox-1 -d 192.168.70.60/test/busybox:latest \ |
| sh -c "while true; do sleep 3600; done" # 添加无限循环保持运行 |

* **思路**：容器需有前台进程，否则会退出（可通过 docker ps -a 查看状态）。

**3. 实验思考**

**实验原理分析**：

* **镜像仓库的作用**：Harbor 作为私有仓库，解决了镜像集中管理、权限控制的问题，相比公共仓库（如 Docker Hub）更安全。
* **标签的意义**：docker tag 本质是创建指向同一镜像 ID 的别名，便于推送至不同仓库。