第二周作业

1. 基于dockerfile，实现分层构建的nginx业务镜像
2. 基于docker实现对容器的CPU和内存的资源限制
3. 部署http协议的harbor镜像仓库
4. 基于docker-compose实现对nginx+tomcat web服务的单机编排
5. 扩展作业：

5.1 掌握containerd的安装

5.2 基于nerdctl拉取镜像和创建容器

1. 基于dockerfile，实现分层构建的nginx业务镜像

```
#清除历史容器
root@castillo:~# docker stop `docker ps | awk '{if (NR>1){print $1}}'`
root@castillo:~# docker rm `docker ps -a | awk '{if (NR>1){print $1}}'`
```

1.1. 编写二进制安装nginxdockerfile

```
FROM ubuntu:22.04
MAINTAINER "jinliucastillo"

ADD sources.list /etc/apt/sources.list

RUN apt update && apt install -y iproute2 ntpdate tcpdump telnet traceroute nfs-kernel-server nfs-common lrzsz tree openssl libssl-dev libpcre3 libpcre3-dev zlib1g-dev ntpdate tcpdump telnet traceroute gcc openssh-server lrzsz tree openssl libssl-dev libpcre3 libpcre3-dev zlib1g-dev ntpdate tcpdump telnet traceroute iotop unzip zip make

ADD nginx-1.22.1.tar.gz /usr/local/src/
RUN cd /usr/local/src/nginx-1.22.1 && ./configure --prefix=/apps/nginx && make && make install && ln -sv /apps/nginx/sbin/nginx /usr/bin
RUN groupadd -g 2088 nginx && useradd -g nginx -s /usr/sbin/nologin -u 2088 nginx && chown -R nginx.nginx /apps/nginx
ADD nginx.conf /apps/nginx/conf/
ADD frontend.tar.gz /apps/nginx/html/

EXPOSE 80 443
CMD ["nginx","-g","daemon off;"]
```

1.2. 构建nginx镜像

```
docker build -t harbor.jinliu.net/myserver/nginx:v1 .
root@castillo:/opt/cicd# docker images
REPOSITORY                         TAG       IMAGE ID       CREATED              SIZE
harbor.jinliu.net/myserver/nginx   v1        5f227fbc596b   About a minute ago   470MB
ubuntu                             22.04     9d28ccdc1fc7   11 months ago        76.3MB
root@castillo:/opt/cicd# docker run -dit -p 80:80 harbor.jinliu.net/myserver/nginx:v1
```

1.3. 提交镜像

```
root@castillo:/opt/cicd# docker commit -m "nginx image v2" -a "jinliu castillo" -c "EXPOSE 80 443" 2330728e0b6e harbor.jinliu.net/myserver/nginx:v2
sha256:e6954854a75e849fc0cb4164da30f2b5dc52d8ac58344e76014eacaf7de2ef3f
root@castillo:/opt/cicd# docker images
REPOSITORY                         TAG       IMAGE ID       CREATED          SIZE
harbor.jinliu.net/myserver/nginx   v2        e6954854a75e   6 seconds ago    514MB
harbor.jinliu.net/myserver/nginx   v1        5f227fbc596b   21 minutes ago   470MB
ubuntu                             22.04     9d28ccdc1fc7   11 months ago    76.3MB
root@castillo:/opt/cicd#
```

2. 基于docker实现对容器的CPU和内存的资源限制

🔷对于Linux 主机，如果没有足够的内容来执行其他重要的系统任务，将会抛出 OOM (Out of Memory Exception,内存溢出、内存泄漏、内存异常), 随后系统会开始杀死进程以释放内存，凡是运行在宿主机的进程都有可能被 kill，包括Dockerd和其它的应用程序，如果重要的系统进程被Kill,会导致和该进程相关的服务全部宕机。

```
Oct 10 17:09:28 docker-server1 kernel: [57402.792264] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=systemd-
journald.service,mems_allowed=0,global_o
om,task_memcg=/system.slice/docker-81aca8d9f526f25eea0015af00271475c9e5a794b762fa70dfe7af2f7efaff64.scope,task=stress-ng-
vm,pid=41298,uid=0
Oct 10 17:09:28 docker-server1 kernel: [57402.792303] Out of memory: Killed process 41298 (stress-ng-vm) total-vm:530572kB, anon-rss:314588kB,
file-rss:4kB, shme
m-rss:24kB, UID:0 pgtables:668kB oom_score_adj:1000
Oct 10 17:09:29 docker-server1 kernel: [57403.278032] sshd invoked oom-killer: gfp_mask=0x1100cca(GFP_HIGHUSER_MOVABLE), order=0,
oom_score_adj=0
Oct 10 17:09:29 docker-server1 kernel: [57403.278038] CPU: 0 PID: 39756 Comm: sshd Not tainted 5.15.0-43-generic #46-Ubuntu
Oct 10 17:09:29 docker-server1 kernel: [57403.278040] Hardware name: VMware, Inc. VMware Virtual Platform/440BX Desktop Reference Platform, BIOS
6.00 07/22/2020
Oct 10 17:09:29 docker-server1 kernel: [57403.278041] Call Trace:
Oct 10 17:09:29 docker-server1 kernel: [57403.278042]  <TASK>
Oct 10 17:09:29 docker-server1 kernel: [57403.278044]  show_stack+0x52/0x58
Oct 10 17:09:29 docker-server1 kernel: [57403.278048]  dump_stack_lvl+0x4a/0x5f
Oct 10 17:09:29 docker-server1 kernel: [57403.278050]  dump_stack+0x10/0x12
Oct 10 17:09:29 docker-server1 kernel: [57403.278052]  dump_header+0x53/0x224
Oct 10 17:09:29 docker-server1 kernel: [57403.278054]  oom_kill_process.cold+0xb/0x10
Oct 10 17:09:29 docker-server1 kernel: [57403.278055]  out_of_memory+0x106/0x2e0
Oct 10 17:09:29 docker-server1 kernel: [57403.278058]  __alloc_pages_slowpath.constprop.0+0x97a/0xa40
```

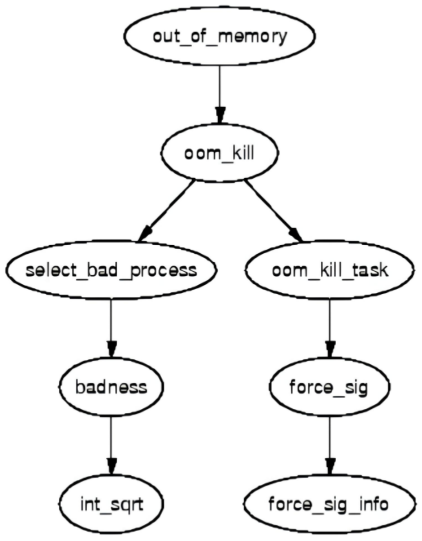
```
#查看内核支持情况
root@castillo:/opt/cicd# docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Docker Buildx (Docker Inc., v0.9.1-docker)
  compose: Docker Compose (Docker Inc., v2.10.2)
  scan: Docker Scan (Docker Inc., v0.17.0)

Server:
 Containers: 4
```

```
   Running: 1
   Paused: 0
   Stopped: 3
  Images: 12
  Server Version: 20.10.18
  Storage Driver: overlay2
   Backing Filesystem: extfs
   Supports d_type: true
   Native Overlay Diff: true
   userxattr: false
  Logging Driver: json-file
  Cgroup Driver: systemd
  Cgroup Version: 2
  Plugins:
   Volume: local
   Network: bridge host ipvlan macvlan null overlay
   Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
  Swarm: inactive
  Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
  Default Runtime: runc
  Init Binary: docker-init
  containerd version: 9cd3357b7fd7218e4aec3eae239db1f68a5a6ec6
  runc version: v1.1.4-0-g5fd4c4d
  init version: de40ad0
  Security Options:
   apparmor
   seccomp
    Profile: default
   cgroupns
  Kernel Version: 5.15.0-52-generic
  Operating System: Ubuntu 22.04.1 LTS
  OSType: linux
  Architecture: x86_64
  CPUs: 2
  Total Memory: 3.832GiB
  Name: castillo
  ID: YUYJ:XCZZ:VP76:ENLD:AFBF:PUR7:2WMK:ENSP:VMYB:WHWT:JKCL:MFNL
  Docker Root Dir: /var/lib/docker
  Debug Mode: false
  Registry: https://index.docker.io/v1/
  Labels:
  Experimental: false
  Insecure Registries:
   127.0.0.0/8
  Registry Mirrors:
   https://r1yv1i78.mirror.aliyuncs.com/
  Live Restore Enabled: false

root@castillo:/opt/cicd#
```

2.1. oom优先级机制

◆OOM优先级机制:

- linux会为每个进程算一个分数，最终他会将分数最高的进程kill。
- /proc/PID/oom_score_adj #范围为-1000到1000，值越高越容易被宿主机kill掉，如果将该值设置为-1000，则进程永远不会被宿主机kernel kill。
- /proc/PID/oom_adj #范围为-17到+15，取值越高越容易被干掉，如果是-17，则表示不能被kill，该设置参数的存在是为了和旧版本的Linux内核兼容。
- /proc/PID/oom_score #这个值是系统综合进程的内存消耗量、CPU时间(utime + stime)、存活时间(uptime – start time)和oom_adj计算出的进程得分，消耗内存越多得分越高，越容易被宿主机kernel强制杀死。



◆物理内存限制参数:

- -m or --memory #限制容器可以使用的最大内存量，如果设置此选项，最小存值为4m（4兆字节）。
- --memory-swap #容器可以使用的交换分区大小，必须要在设置了物理内存限制的前提才能设置交换分区的限制
- --memory-swappiness #设置容器使用交换分区的倾向性，值越高表示越倾向于使用swap分区，范围为0-100，0为能不用就不用，100为能用就用。
- --kernel-memory #容器可以使用的最大内核内存量，最小为4m，由于内核内存与用户空间内存隔离，因此无法与用户空间内存直接交换，因此内核内存不足的容器可能会阻塞宿主主机资源，这会对主机和其他容器或者其他服务进程产生影响，因此不要设置内核内存大小。
- --memory-reservation #允许指定小于--memory的软限制，当Docker检测到主机上的争用或内存不足时会激活该限制，如果使用--memory-reservation，则必须将其设置为低于--memory才能使其优先。 因为它是软限制，所以不能保证容器不超过限制。
- --oom-kill-disable #默认情况下，发生OOM时，kernel会杀死容器内进程，但是可以使用--oom-kill-disable参数，可以禁止oom发生在指定的容器上，即 仅在已设置-m / - memory选项的容器上禁用OOM，如果-m 参数未配置，产生OOM时，主机为了释放内存还会杀死系统进程。

2.2. 测试内存限制

```
root@castillo:/opt/cicd# docker run -it --rm --name jinliu-c1 lorel/docker-stress-ng --vm 2 --vm-bytes 256M
stress-ng: info: [1] defaulting to a 86400 second run per stressor
stress-ng: info: [1] dispatching hogs: 2 vm

root@castillo:~# docker stats
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O
   BLOCK I/O     PIDS
5dd8008128b8   jinliu-c1         198.06%   515.8MiB / 3.832GiB   13.15%    796B / 0B
   0B / 0B       5
2330728e0b6e   sleepy_johnson    0.00%     15.47MiB / 3.832GiB   0.39%     11MB /
110kB   0B / 121MB   2

root@castillo:~# docker run -it --rm -m 256m --name jinliu-c2 lorel/docker-stress-ng --vm 2 --vm-bytes 256M
Given value bytes is not a valid decimal for the vm option

root@castillo:~
```

```
root@castillo:~# docker run -it --rm -m 256m --name jinliu-c2 lorel/docker-stress-ng
--vm 2 --vm-bytes 256M
stress-ng: info: [1] defaulting to a 86400 second run per stressor
stress-ng: info: [1] dispatching hogs: 2 vm


root@castillo:~# docker stats
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O
   BLOCK I/O          PIDS
2330728e0b6e   sleepy_johnson    0.00%     15.47MiB / 3.832GiB   0.39%     11MB /
110kB   0B / 121MB        2
716e0f480798   jinliu-c2         17.54%    255.9MiB / 256MiB     99.95%    656B / 0B
   99.5MB / 3.86GB   5
```

2.3. 测试cpu限制

✓ 注：CPU资源限制是将分配给容器的2核心分配到了宿主机每一核心CPU上，也就是容器的总CPU值是在宿主机的每一个核心CPU分配了部分比例。

- Tasks: 288 total, 10 running, 278 sleeping,   0 stopped,   0 zombie
- %Cpu0 : 51.2 us, 0.0 sy, 0.0 ni, 48.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
- %Cpu1 : 26.4 us, 23.4 sy, 0.0 ni, 50.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
- %Cpu2 : 23.1 us, 27.7 sy, 0.0 ni, 49.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
- %Cpu3 : 18.3 us, 31.3 sy, 0.0 ni, 50.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

```
root@castillo:~# docker run -it --rm --name jjinliu-c1 --cpus 2 lorel/docker-stress-ng
--cpu 4 --vm 4
stress-ng: info: [1] defaulting to a 86400 second run per stressor
stress-ng: info: [1] dispatching hogs: 4 cpu, 4 vm
^Cstress-ng: info: [1] successful run completed in 91.84s
root@castillo:~#
```

2.4. 将容器运行到指定的CPU上

```
root@castillo:~# docker run -it --rm --name jinliu-c1 --cpus 1  --cpuset-cpus 1
lorel/docker-stress-ng --cpu 2 --vm 2
stress-ng: info: [1] defaulting to a 86400 second run per stressor
stress-ng: info: [1] dispatching hogs: 2 cpu, 2 vm

root@castillo:~# docker stats
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O
   BLOCK I/O     PIDS
2330728e0b6e   sleepy_johnson    0.00%     15.47MiB / 3.832GiB   0.39%     11MB /
110kB   0B / 121MB   2
6fbae7fcc895   jinliu-c1         100.35%   519.9MiB / 3.832GiB   13.25%    726B / 0B
   0B / 0B       7

root@castillo:/opt/cicd# top
top - 17:08:28 up 10 days,  2:41,  8 users,  load average: 0.67, 1.56, 1.21
Tasks: 140 total,   5 running, 135 sleeping,   0 stopped,   0 zombie
%Cpu0  :  0.3 us,  0.3 sy,  0.0 ni, 99.0 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st

%Cpu1  : 85.4 us, 14.6 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
MiB Mem :   3924.0 total,   1197.7 free,    587.1 used,   2139.2 buff/cache
MiB Swap:   3925.0 total,   3924.7 free,      0.3 used.   3044.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  32574 root      20   0    6908   2496    672 R  25.0   0.1   0:01.65 stress-ng-cpu
  32576 root      20   0    6908   2496    672 R  25.0   0.1   0:01.66 stress-ng-cpu
  32578 root      20   0  268408 150288    540 R  25.0   3.7   0:01.65 stress-ng-vm
  32579 root      20   0  268408 152136    540 R  25.0   3.8   0:01.65 stress-ng-vm
```

2.5. 对比两个容器cpu限制情况

```
#C1
root@castillo:~# docker run -it --rm --name jinliu-c1 --cpus 1  --cpu-shares 800 lorel/docker-stress-ng --cpu 1 --vm 2
stress-ng: info: [1] defaulting to a 86400 second run per stressor
stress-ng: info: [1] dispatching hogs: 1 cpu, 2 vm

root@castillo:~# docker stats
CONTAINER ID   NAME             CPU %     MEM USAGE / LIMIT     MEM %     NET I/O
   BLOCK I/O    PIDS
2330728e0b6e   sleepy_johnson   0.00%     15.47MiB / 3.832GiB   0.39%     11MB /
110kB   0B / 121MB   2
5640562a51c1   jinliu-c1        97.83%    517.8MiB / 3.832GiB   13.20%    796B / 0B
   0B / 0B      6

root@castillo:~# top
top - 17:13:30 up 10 days,  2:46, 10 users,  load average: 1.23, 1.55, 1.34
Tasks: 147 total,   5 running, 142 sleeping,   0 stopped,   0 zombie
%Cpu(s): 51.0 us,  0.5 sy,  0.0 ni, 48.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3924.0 total,    997.4 free,    787.3 used,   2139.3 buff/cache
MiB Swap:   3925.0 total,   3924.7 free,      0.3 used.   2844.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  32745 root      20   0  268408 262544    596 R  49.5   6.5   0:37.75 stress-ng-vm
  32746 root      20   0  268408 262544    596 R  25.2   6.5   0:18.99 stress-ng-vm
  32742 root      20   0    6908   2432    604 R  24.9   0.1   0:19.01 stress-ng-cpu
  2596 root      20   0 1356280  58640  35812 S   0.3   1.5   1:34.86 containerd
 10621 castillo  20   0   17300   8072   5660 S   0.3   0.2   0:01.02 sshd

#C2
root@castillo:~# docker run -it --rm --name jinliu-c2 --cpus 1  --cpu-shares 400 lorel/docker-stress-ng --cpu 1 --vm 2
stress-ng: info: [1] defaulting to a 86400 second run per stressor
stress-ng: info: [1] dispatching hogs: 1 cpu, 2 vm

root@castillo:~# docker stats
2330728e0b6e   sleepy_johnson   0.00%     15.47MiB / 3.832GiB   0.39%     11MB / 110kB   0B / 121MB   2
5640562a51c1   jinliu-c1        100.22%   517.8MiB / 3.832GiB   13.20%    866B / 0B      0B / 0B      6
03dd3139eb2d   jinliu-c2        98.85%    314.3MiB / 3.832GiB   8.01%     656B / 0B      0B / 0B      6

root@castillo:~# top1
top - 17:15:15 up 10 days,  2:47, 10 users,  load average: 3.74, 2.09, 1.54
Tasks: 153 total,   7 running, 146 sleeping,   0 stopped,   0 zombie
%Cpu(s): 99.0 us,  0.7 sy,  0.0 ni,  0.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3924.0 total,    466.8 free,   1315.5 used,   2141.7 buff/cache
MiB Swap:   3925.0 total,   3924.7 free,      0.3 used.   2314.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  32831 root      20   0  268408 262528    580 R  39.3   6.5   0:15.76 stress-ng-vm
  32742 root      20   0    6908   2432    604 R  37.7   0.1   0:50.53 stress-ng-cpu
  32746 root      20   0  268408 262544    596 R  32.3   6.5   0:48.67 stress-ng-vm
  32745 root      20   0  268408 262544    596 R  30.0   6.5   1:21.84 stress-ng-vm
  32828 root      20   0    6908   2456    628 R  29.3   0.1   0:12.53 stress-ng-cpu
  32832 root      20   0  268408 262528    580 R  29.0   6.5   0:12.90 stress-ng-vm
   4154 root      20   0 1595372  84380  50980 S   0.3   2.1   1:29.97 dockerd
  31232 root      20   0  712656  11152   8296 S   0.3   0.3   0:03.21 containerd-
```

```
shim
```

2.6. systemd验证

```
root@castillo:~# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
         STATUS          PORTS                                        NAMES
5565cf04c53b   lorel/docker-stress-ng               "/usr/bin/stress-ng …"   38
seconds ago   Up 36 seconds                                                jinliu-c1
2330728e0b6e   harbor.jinliu.net/myserver/nginx:v1   "nginx -g 'daemon of…"   2 hours
ago       Up 2 hours      0.0.0.0:80->80/tcp, :::80->80/tcp, 443/tcp   sleepy_johnson
root@castillo:~# ps -ef |grep nginx
root       31251   31232  0 15:31 pts/0    00:00:00 nginx: master process nginx -g
daemon off;
nobody     31505   31251  0 15:39 pts/0    00:00:00 nginx: worker process
root       33005   32653  0 17:21 pts/8    00:00:00 grep --color=auto nginx
root@castillo:~# cat /proc/31251/cpuset
/system.slice/docker-
2330728e0b6e9110151f74b714365945522e66e8c03d723f875e09c112748d6c.scope
root@castillo:~# cat /sys/fs/cgroup/system.slice/docker-
2330728e0b6e9110151f74b714365945522e66e8c03d723f875e09c112748d6c.scope/cpu.max
max 100000
root@castillo:~# cat /sys/fs/cgroup/system.slice/docker-
2330728e0b6e9110151f74b714365945522e66e8c03d723f875e09c112748d6c.scope/memory.max
max
root@castillo:~#
```

3. 部署http协议的harbor镜像仓库

3.1. 优点

➢基于角色的访问控制：用户与Docker镜像仓库通过“项目”进行项目管理，可以对不同的账户设置不同的权限，以实现权限的精细管控。

➢镜像复制：镜像可以在多个Registry实例中复制（同步），可以实现高性能、高可用的镜像服务。

➢图形化用户界面：用户可以通过浏览器来浏览，管理当前Docker镜像仓库，管理项目和镜像等。

➢AD/LDAP 支：Harbor可以集成企业内部已有的AD/LDAP，用于鉴权认证管理。

➢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。

➢国际化：已拥有英文、中文、德文、日文和俄文等多语言支持版本。

➢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。

➢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。

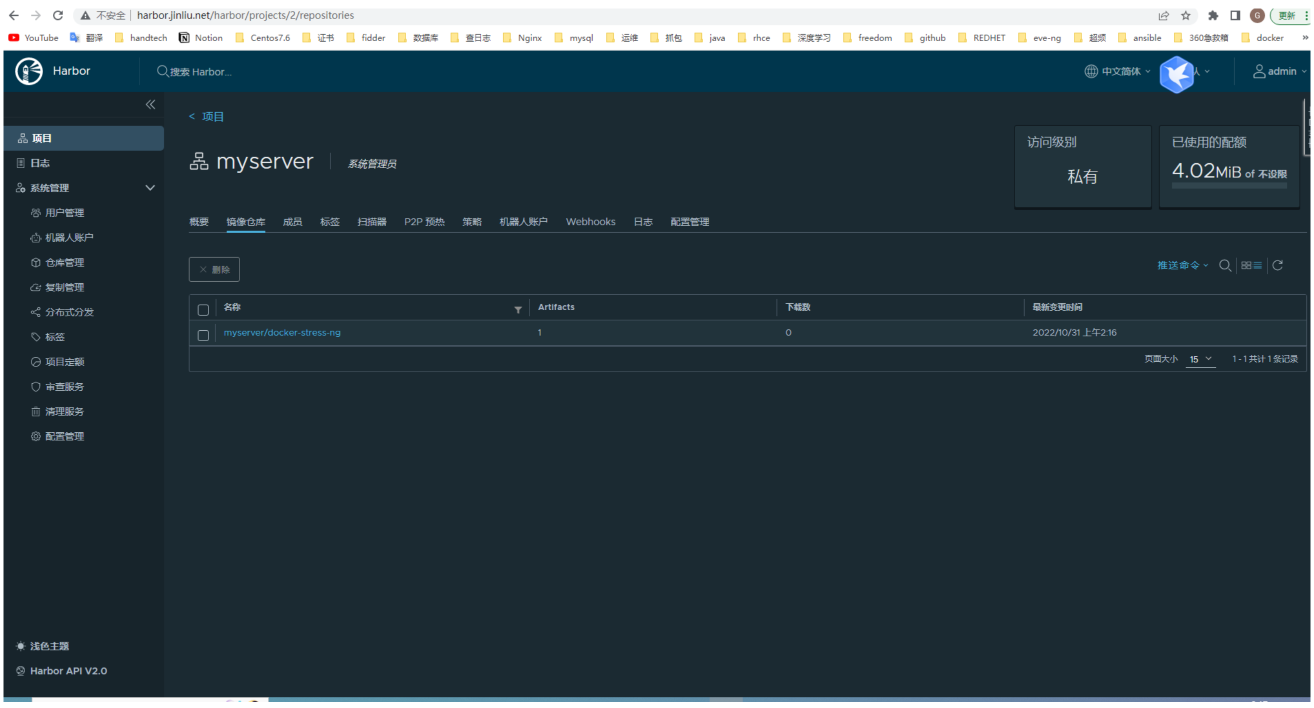
3.2. 安装harbor

```
#修改域名
root@castillo:/apps# ls -l
total 739540
-rw-rw-r-- 1 castillo castillo 757278514 Oct 30 14:48 harbor-offline-installer-v2.6.1.tgz
drwxr-xr-x 2 root     root          4096 Oct 30 14:30 nginx
root@castillo:/apps# tar -zxf harbor-offline-installer-v2.6.1.tgz
root@castillo:/apps# cd harbor
root@castillo:/apps/harbor# vim harbor.yml
root@castillo:/apps/harbor# grep -A 10 "jinliu" harbor.yml
hostname: harbor.jinliu.net

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

# https related config
https:
  # https port for harbor, default is 443
  port: 443
root@castillo:/apps/harbor#

#配置本地域名解析
root@castillo:/apps/harbor# cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 castillo

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

192.168.31.113 harbor.jinliu.net
root@castillo:/apps/harbor#

#配置域名信任
root@castillo:/apps/harbor# cat /etc/docker/daemon.json
{
  "registry-mirrors": ["https://r1yv1i78.mirror.aliyuncs.com"],
  "insecure-registries": ["harbor.jinliu.net","192.168.31.113"]
}
root@castillo:/apps/harbor#

#安装
root@castillo:/apps/harbor# bash install.sh

#登录
root@castillo:/apps/harbor# docker login harbor.jinliu.net
Username: admin
```

```
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@castillo:/apps/harbor#
```

3.3. 推进镜像

```
root@castillo:/apps/harbor# docker tag lorel/docker-stress-ng:latest
harbor.jinliu.net/myserver/docker-stress-ng:latest
root@castillo:/apps/harbor# docker push  harbor.jinliu.net/myserver/docker-stress-
ng:latest
The push refers to repository [harbor.jinliu.net/myserver/docker-stress-ng]
5f70bf18a086: Pushed
ea580b0285fe: Pushed
6102f0d2ad33: Pushed
latest: digest:
sha256:5768a5d8e196be8c8fabbda04d937aabe1407f397b2f12e1fea2573e4b9d9bef size: 1563
root@castillo:/apps/harbor#
root@castillo:/apps/harbor#
root@castillo:/apps/harbor#
root@castillo:/apps/harbor#
```


不安全 | harbor.jinliu.net/harbor/projects/2/repositories
YouTube 翻译 handtech Notion Centos7.6 证书 fidder 数据库 查日志 Nginx mysql 运维 抓包 java rhce 深度学习 freedom github REDHET eve-ng 超频 ansible 360急救箱 docker
Harbor
搜索 Harbor...
中文简体
admin
项目
日志
系统管理
用户管理
机器人账户
仓库管理
复制管理
分布式分发
标签
项目定额
审查服务
清理服务
配置管理
浅色主题
Harbor API V2.0
< 项目
myserver
系统管理员
访问级别
私有
已使用的配额
4.02MiB of 不设限
概要 镜像仓库 成员 标签 扫描器 P2P 预热 策略 机器人账户 Webhooks 日志 配置管理
删除
推送命令

名称	Artifacts	下载数	最新变更时间
myserver/docker-stress-ng	1	0	2022/10/31 上午2:16

页面大小 15　1 - 1 共计 1 条记录