Docker 容器技术学习

葛久平

2022 年 3 月 30 日

长安数字工作室

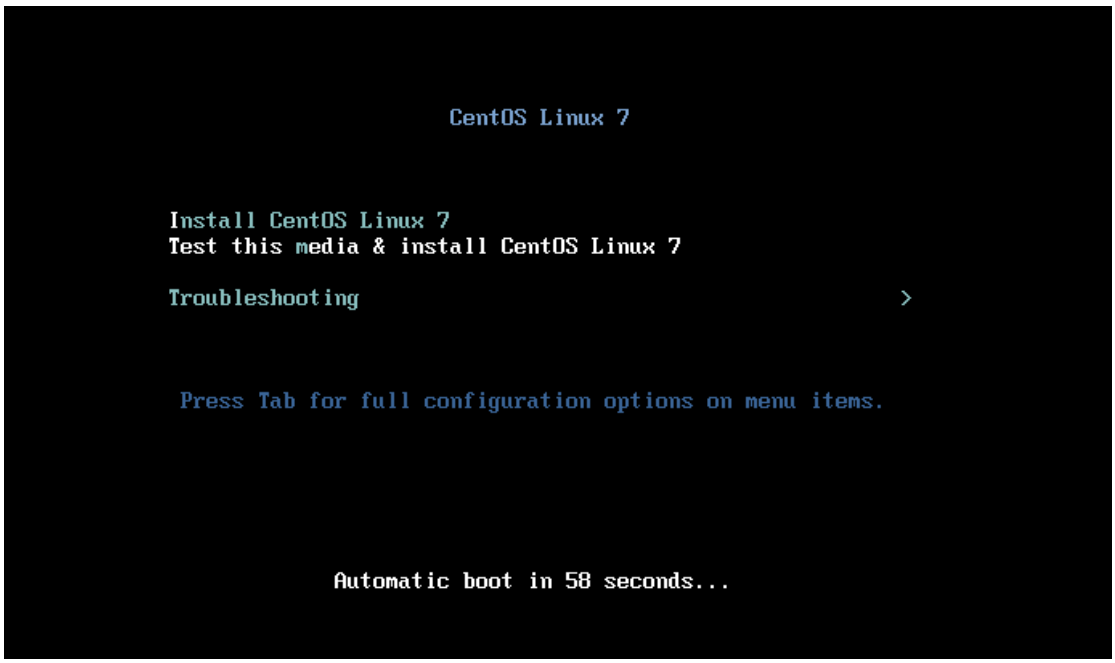
目 录

一、安装 Docker

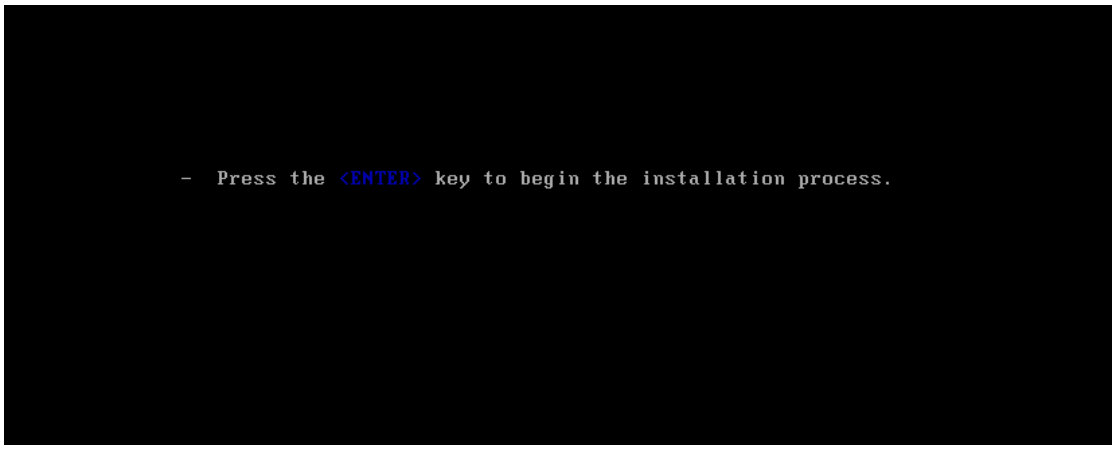
（一）安装 centos

安装 centos7.3 以上最小化系统（以 centos7.3（1611）安装为例）

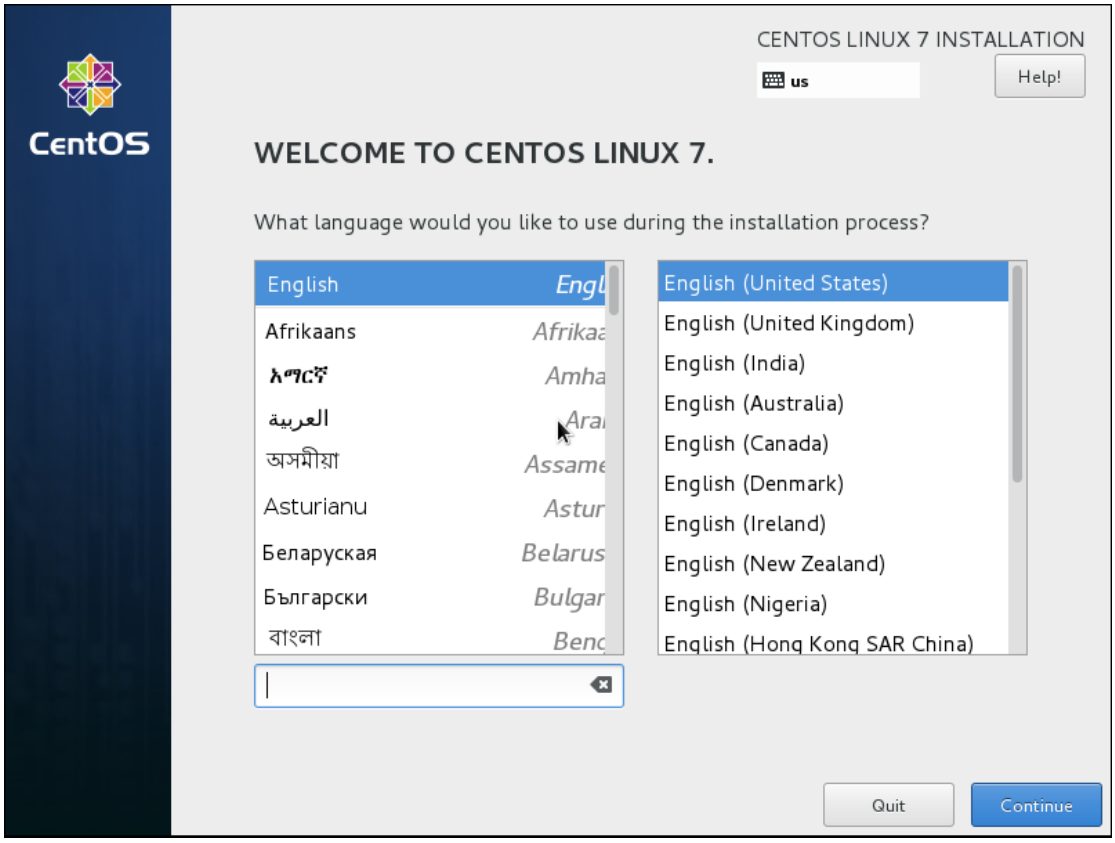
1.选择第一项“Install Centos Linux7”



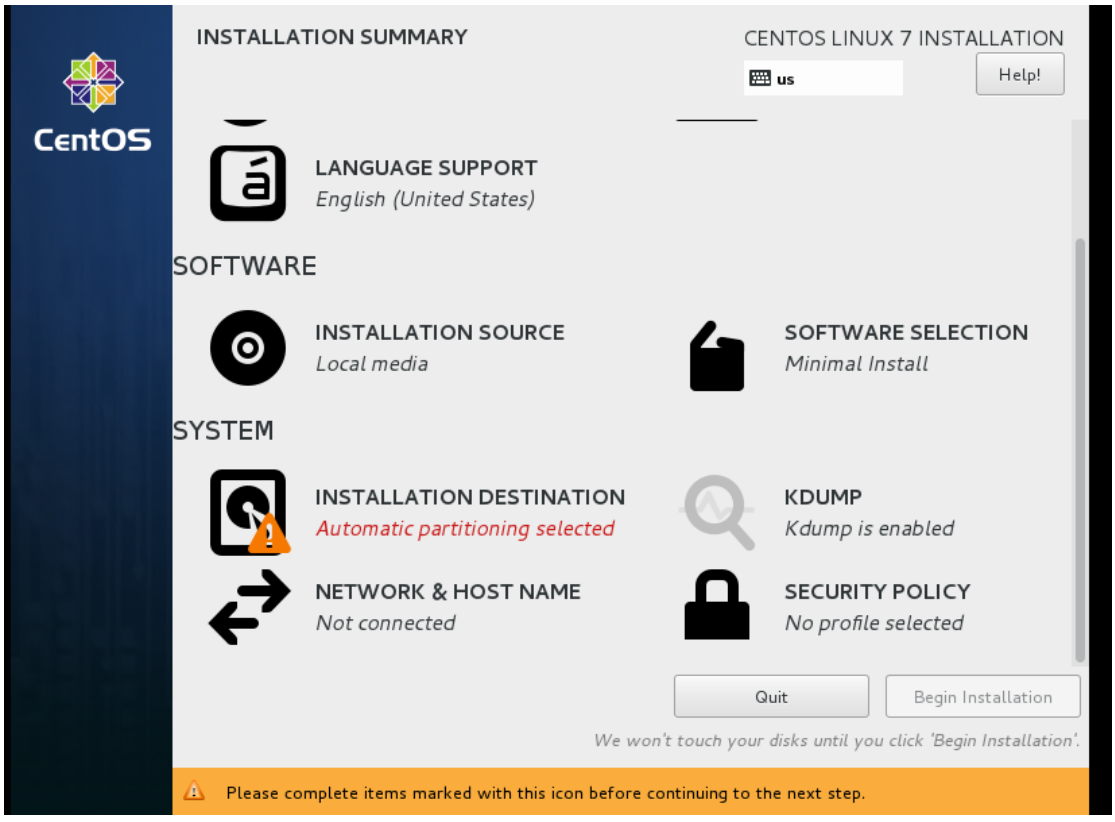
2.回车



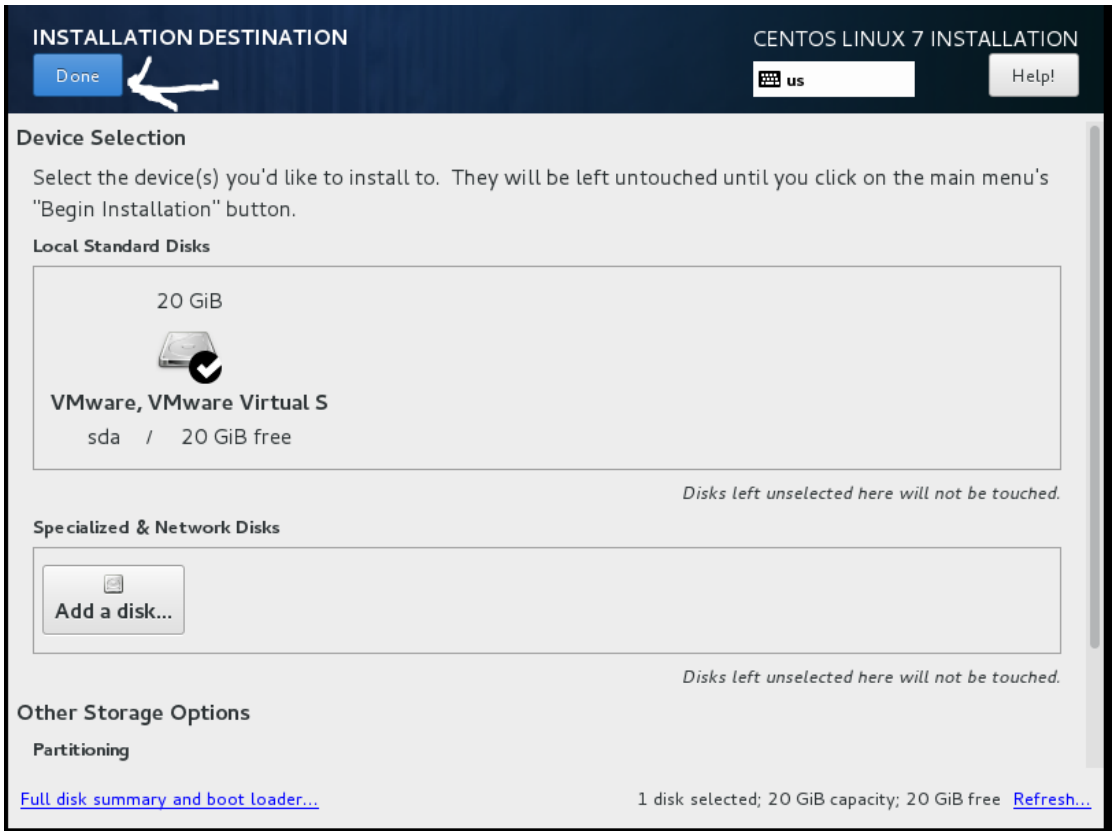
3.默认，单击 continue



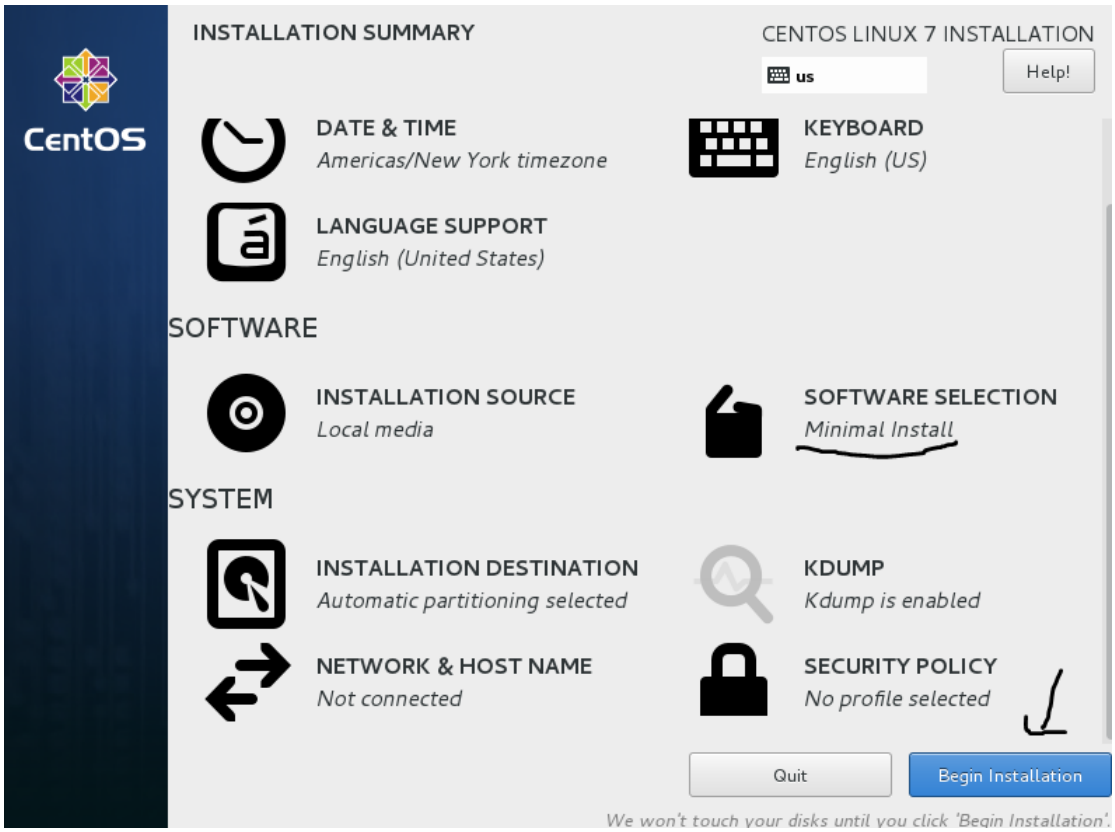
4.单击 system 项，设置安装磁盘



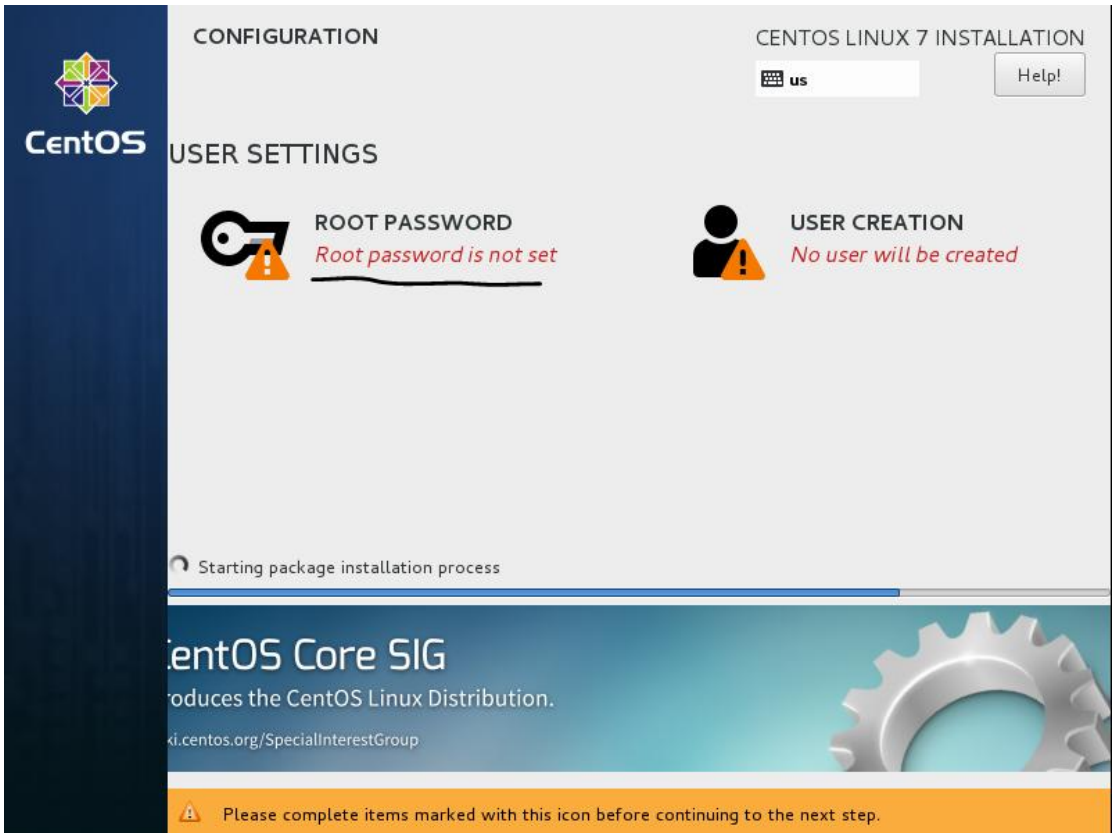
5.单击 done



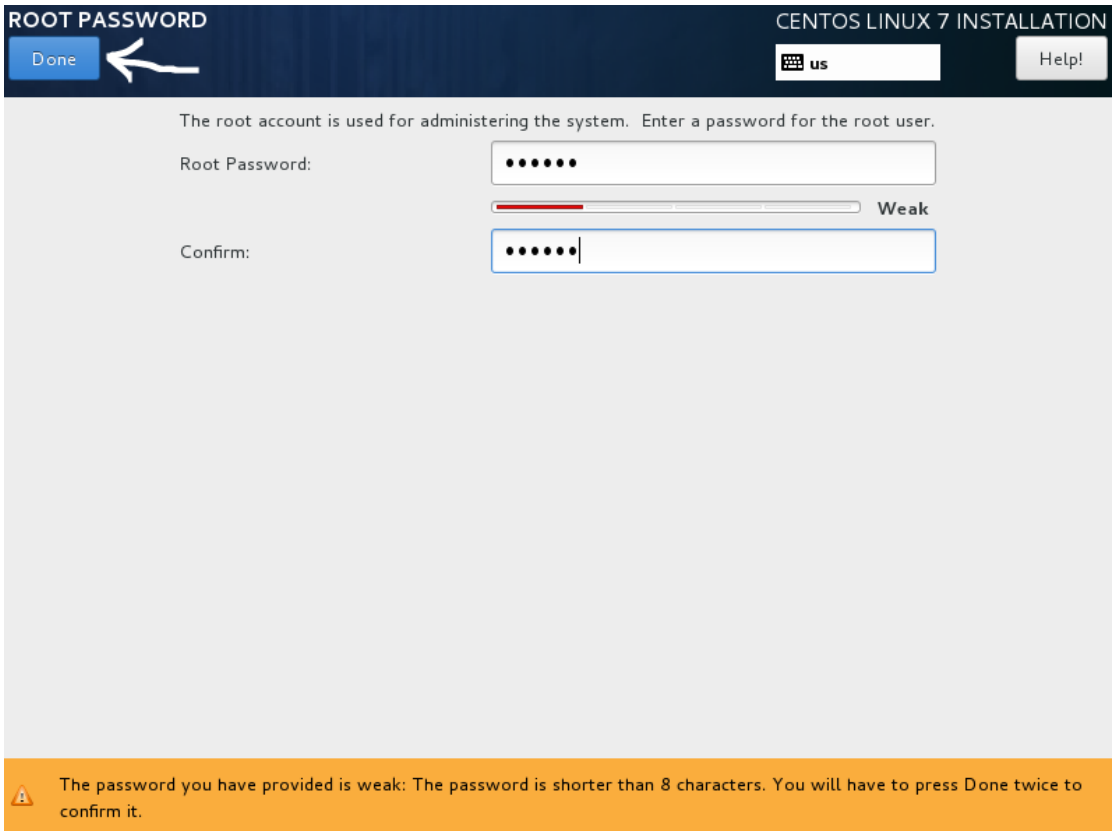
6.单击 Begin Installation



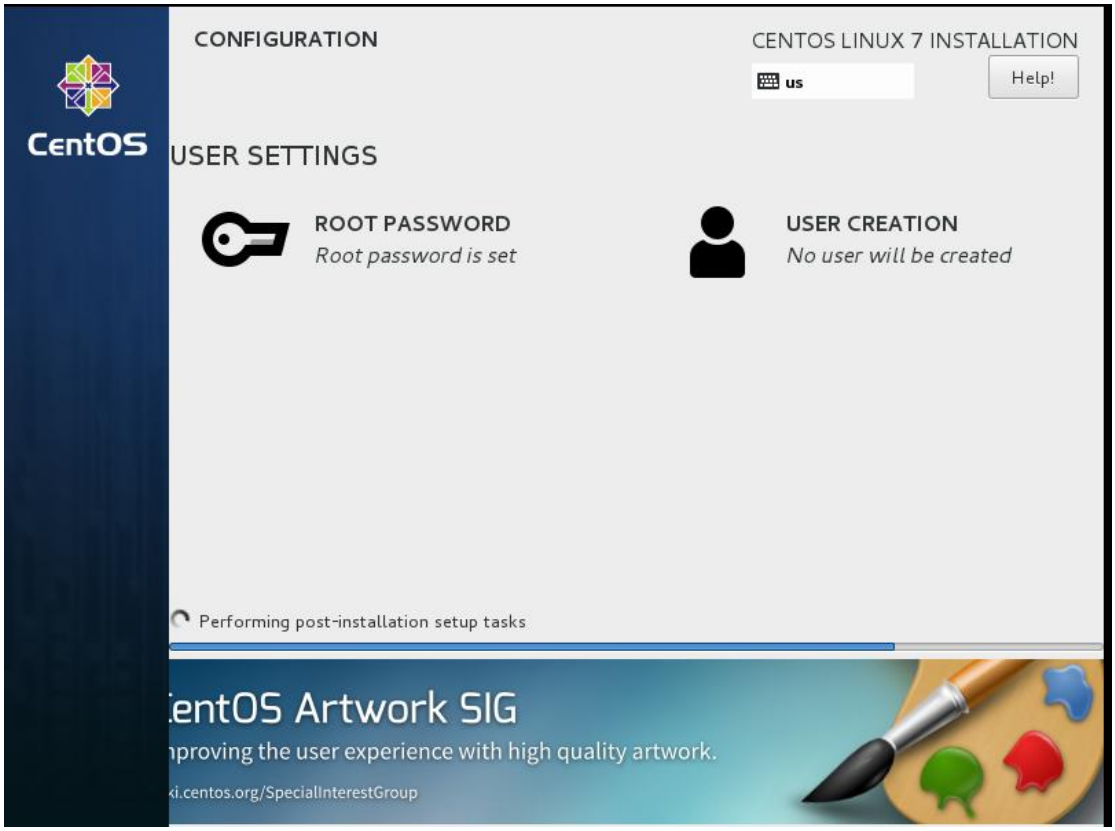
7.单击 USER SETTIONS 设置 root 密码



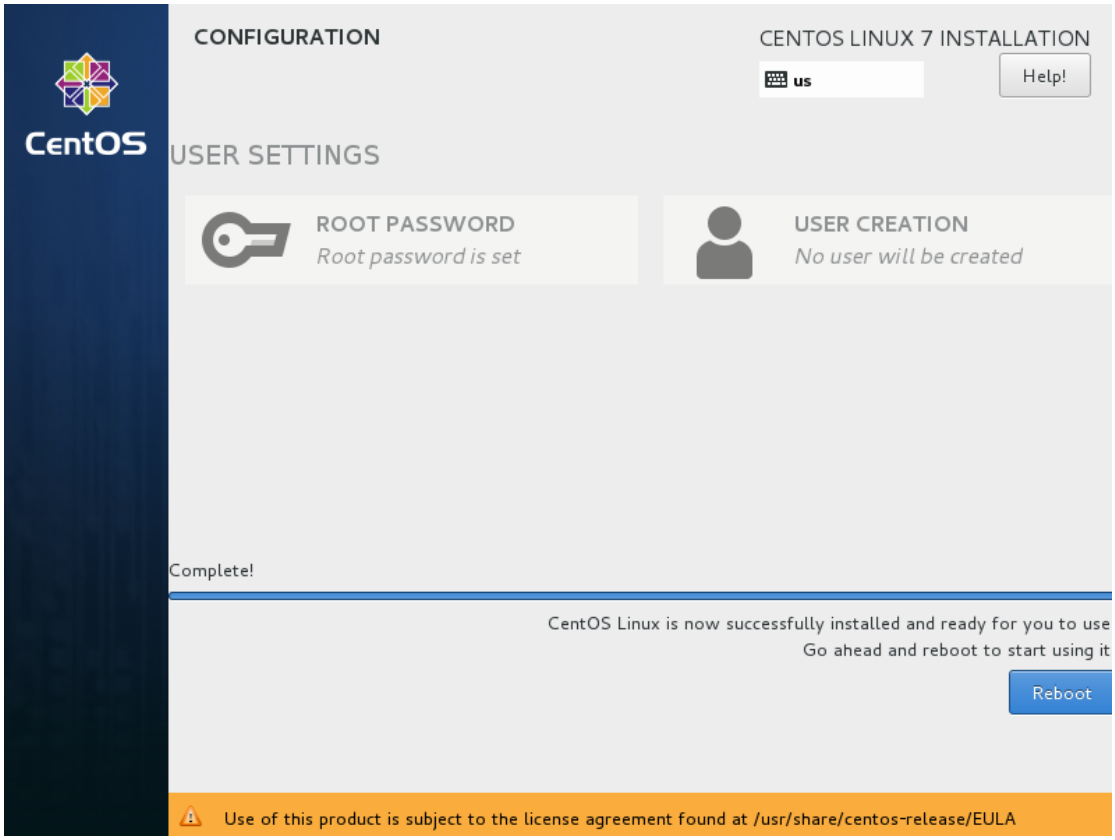
8.设置 root 密码



9.安装



10.单击 REBOOT，重新启动系统



11.进入系统

```
CentOS Linux 7 (Core)
Kernel 3.10.0-514.el7.x86_64 on an x86_64

localhost login: root
Password:
[root@localhost ~]# _
```

11.设置 IP 属性，保证该系统能够上网

 # vi /etc/sysconfig/network-scripts/ifcfg-ens33

```
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens33
UUID=e4364019-c4ea-4472-ad7d-98205738c2ba
DEVICE=ens33
ONBOOT=yes
IPADDR=192.168.200.10
NETMASK=255.255.255.0
GATEWAY=192.168.200.2
DNS1=8.8.8.8_
```

12.重新启动网络服务，让 IP 属性设置生效

#systemctl restart network

13.查看 IP 属性设置

 # ip a

```
[root@localhost ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:d1:51:fc brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.10/24 brd 192.168.200.255 scope global noprefixroute ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::29f5:d088:c57a:8fa1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

14.因特网连通性测试

 # ping www.baidu.com

```
[root@localhost ~]# ping www.baidu.com
PING www.a.shifen.com (61.135.169.125) 56(84) bytes of data.
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=1 ttl=128 time=3.88 ms
From 192.216.5.108 (192.216.5.108) icmp_seq=16 Destination Host Unreachable
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=17 ttl=128 time=5.44 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=18 ttl=128 time=4.32 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=19 ttl=128 time=4.65 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=20 ttl=128 time=4.22 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=21 ttl=128 time=5.74 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=22 ttl=128 time=4.70 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=23 ttl=128 time=6.21 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=24 ttl=128 time=5.55 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=25 ttl=128 time=5.04 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=26 ttl=128 time=5.00 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=27 ttl=128 time=5.84 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=28 ttl=128 time=4.53 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=29 ttl=128 time=5.50 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=30 ttl=128 time=5.73 ms
64 bytes from 61.135.169.125 (61.135.169.125): icmp_seq=31 ttl=128 time=4.72 ms
```

（二）查看 centos 内核版本

确保系统核心版本在 3.10 以上

#uname –sr

```
[root@openstack ~]# uname -sr
Linux 3.10.0-1062.1.2.el7.x86_64
[root@openstack ~]#
```

（三）yum 包更新到最新

yum update

centos 镜像选择 1611 以上，此步骤不用做

yum -y update：升 bai 级所 du 有包同时也升级软件 zhi 和系统内核；

yum -y upgrade：只升级所有包，不升级软件和系统内核。

（四）安装需要的软件包

yum-util 提供 yum-config-manager 功能，另外两个是 devicemapper 驱动依赖的

yum install -y yum-utils device-mapper-persistent-data lvm2

```
[root@centos ~]# yum install -y yum-utils device-mapper-persistent-data lvm2
Loaded plugins: fastestmirror
base                                                     | 3.6 kB     00:00
extras                                                   | 2.9 kB     00:00
updates                                                  | 2.9 kB     00:00
(1/4): base/7/x86_64/group_gz                              | 153 kB   00:00
(2/4): extras/7/x86_64/primary_db                          | 206 kB   00:00
(3/4): updates/7/x86_64/primary_db                         | 4.5 MB   00:00
(4/4): base/7/x86_64/primary_db                            | 6.1 MB   00:22
Determining fastest mirrors
 * base: mirrors.bfsu.edu.cn
 * extras: mirrors.bfsu.edu.cn
 * updates: mirrors.bfsu.edu.cn
Resolving Dependencies
--> Running transaction check
---> Package device-mapper-persistent-data.x86_64 0:0.6.3-1.el7 will be updated
---> Package device-mapper-persistent-data.x86_64 0:0.8.5-2.el7 will be an updat
e
```

（五）设置下载 docker 安装 yum 源

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

上面是官方的源在国外，为了提高下载速度建议使用国内的源，经常使用下面阿里源

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

```
[root@centos ~]#  yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
[root@centos ~]#
```

上面是官方的源在国外，为了提高下载速度建议使用国内的源，**经常使用阿里源：**

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

清华大学源：

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

（六）查看所有仓库中所有 docker 版本，并选择特定版本安装

yum list docker-ce --showduplicates | sort -r

```
[root@openstack ~]# yum list docker-ce --showduplicates |sort -r
 * updates: mirrors.huaweicloud.com
Loading mirror speeds from cached hostfile
Loaded plugins: fastestmirror
Installed Packages
 * extras: mirrors.huaweicloud.com
docker-ce.x86_64            3:19.03.3-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.3-3.el7                     @docker-ce-stable
docker-ce.x86_64            3:19.03.2-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.1-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.0-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.9-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.8-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.7-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.6-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.5-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.4-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.3-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.2-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.1-3.el7                     docker-ce-stable
docker-ce.x86_64            3:18.09.0-3.el7                     docker-ce-stable
docker-ce.x86_64            18.06.3.ce-3.el7                    docker-ce-stable
docker-ce.x86_64            18.06.2.ce-3.el7                    docker-ce-stable
docker-ce.x86_64            18.06.1.ce-3.el7                    docker-ce-stable
docker-ce.x86_64            18.06.0.ce-3.el7                    docker-ce-stable
docker-ce.x86_64            18.03.1.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            18.03.0.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.12.1.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.12.0.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.09.1.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.09.0.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.06.2.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.06.1.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.06.0.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.03.3.ce-1.el7                    docker-ce-stable
docker-ce.x86_64            17.03.2.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.03.1.ce-1.el7.centos             docker-ce-stable
docker-ce.x86_64            17.03.0.ce-1.el7.centos             docker-ce-stable
 * base: mirrors.huaweicloud.com
Available Packages
[root@openstack ~]#
```

（七）安装 docker（ce 是免费版、ee 是收费）

yum install docker-ce　//下载是最新版

```
[root@localhost ~]# yum install docker-ce
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.bit.edu.cn
 * extras: mirrors.huaweicloud.com
 * updates: mirrors.huaweicloud.com
Resolving Dependencies
--> Running transaction check
---> Package docker-ce.x86_64 3:19.03.5-3.el7 will be installed
--> Processing Dependency: container-selinux >= 2:2.74 for package: 3:docker-ce-19.03.5-3.el7.x86_64
--> Processing Dependency: containerd.io >= 1.2.2-3 for package: 3:docker-ce-19.03.5-3.el7.x86_64
--> Processing Dependency: libseccomp >= 2.3 for package: 3:docker-ce-19.03.5-3.el7.x86_64
--> Processing Dependency: docker-ce-cli for package: 3:docker-ce-19.03.5-3.el7.x86_64
--> Processing Dependency: libcgroup for package: 3:docker-ce-19.03.5-3.el7.x86_64
--> Processing Dependency: libseccomp.so.2()(64bit) for package: 3:docker-ce-19.03.5-3.el7.x86_64
--> Running transaction check
```

或者下载指定版本 17.12.0.ce

yum install docker-ce-17.12.0.ce

安装好的 docker 系统有两个程序：docker 服务端和客户端。

（八）启动 docker

#systemctl start docker

```
[root@localhost ~]# systemctl start docker
[root@localhost ~]#
```

（九）设置开机启动 docker

#systemctl enable docker

```
[root@localhost ~]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/d
ocker.service.
[root@localhost ~]#
```

（十）查看 docker 版本

#docker version

```
[root@openstack ~]# systemctl start docker
[root@openstack ~]# docker version
Client: Docker Engine - Community
 Version:           19.03.3
 API version:       1.40
 Go version:        go1.12.10
 Git commit:        a872fc2f86
 Built:             Tue Oct  8 00:58:10 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.3
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.10
  Git commit:       a872fc2f86
  Built:            Tue Oct  8 00:56:46 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.10
  GitCommit:        b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:          1.0.0-rc8+dev
  GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
[root@openstack ~]#
```

（十一）docker 安装方法 2

直接从官网下载安装，下载的是一个脚本直接运行

1.最小化安装 centos，核心 3.10 以上

2.安装

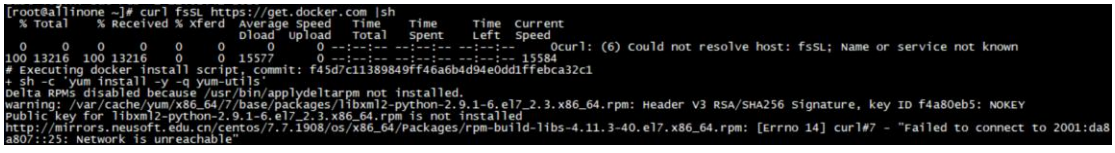
curl fsSL https://get.docker.com |sh

-L 跟随重定向

-S 显示错误信息

-s 静默模式，不输出任何信息

-f 连接失败是不显示 http 错误



3.启动 docker

systemctl start docker

4.查看 docker 版本

docker version

```
[root@allinone ~]# docker version
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea
 Built:             Wed Nov 13 07:25:41 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.5
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.12
  Git commit:       633a0ea
  Built:            Wed Nov 13 07:24:18 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.10
  GitCommit:        b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:          1.0.0-rc8+dev
  GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
[root@allinone ~]#
```

```
CentOS-8 - Base - mirrors.aliyun.com                                    44 kB/s | 3.9 kB     00:00
CentOS-8 - Extras - mirrors.aliyun.com                                  18 kB/s | 1.5 kB     00:00
CentOS-8 - AppStream - mirrors.aliyun.com                               43 kB/s | 4.3 kB     00:00
Docker CE Stable - x86_64                                              7.0 kB/s |  25 kB     00:03
Metadata cache created.
+ '[' -n '' ']'
+ sh -c 'yum install -y -q docker-ce'
Error:
 Problem: package docker-ce-3:19.03.12-3.el7.x86_64 requires containerd.io >= 1.2.2-3, but none of the providers can be installed
  - cannot install the best candidate for the job
  - package containerd.io-1.2.10-3.2.el7.x86_64 is filtered out by modular filtering
  - package containerd.io-1.2.13-3.1.el7.x86_64 is filtered out by modular filtering
  - package containerd.io-1.2.13-3.2.el7.x86_64 is filtered out by modular filtering
  - package containerd.io-1.2.2-3.3.el7.x86_64 is filtered out by modular filtering
  - package containerd.io-1.2.2-3.el7.x86_64 is filtered out by modular filtering
  - package containerd.io-1.2.4-3.1.el7.x86_64 is filtered out by modular filtering
  - package containerd.io-1.2.5-3.1.el7.x86_64 is filtered out by modular filtering
  - package containerd.io-1.2.6-3.3.el7.x86_64 is filtered out by modular filtering
[root@centOs82 ~]# dnf install https://download.docker.com/linux/centos/7/x86_64/stable/Packages/containerd.io-1.2.6-3.3.el7.x86_64.rpm
Last metadata expiration check: 0:01:04 ago on Mon 24 Aug 2020 11:30:30 PM EDT.
containerd.io-1.2.6-3.3.el7.x86_64.rpm                    34% [=================-          ]  42 kB/s | 8.8 MB     06:56
```

dnf install https://download.docker.com/linux/centos/7/x86_64/stable/Packages/containerd.io-1.2.6-3.3.el7.x86_64.rpm

（十二）docker 安装方法 3

1.最小化安装 centos，核心 3.10 以上

2.安装

curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun

```
[root@allinone ~]# curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun
# Executing docker install script, commit: f45d7c11389849ff46a6b4d94e0dd1ffebca32c1
+ sh -c 'yum install -y -q yum-utils'
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
warning: /var/cache/yum/x86_64/7/base/packages/libxml2-python-2.9.1-6.el7_2.3.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID f4a80eb5: NOKEY
Public key for libxml2-python-2.9.1-6.el7_2.3.x86_64.rpm is not installed
Importing GPG key 0xF4A80EB5:
 Userid     : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"
 Fingerprint: 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5
 Package    : centos-release-7-2.1511.el7.centos.2.10.x86_64 (@anaconda)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
+ sh -c 'yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo'
```

3.启动 docker

systemctl start docker

4.查看 docker 版本

docker version

```
[root@allinone ~]# docker version
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea
 Built:             Wed Nov 13 07:25:41 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.5
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.12
  Git commit:       633a0ea
  Built:            Wed Nov 13 07:24:18 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.10
  GitCommit:        b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:          1.0.0-rc8+dev
  GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
[root@allinone ~]#
```

curl -sSL https://get.daocloud.io/docker | sh

```
[root@centos ~]# curl -sSL https://get.daocloud.io/docker | sh
# Executing docker install script, commit: 26ff363bcf3b3f5a00498ac43694bf1c7d9ce16c
+ sh -c 'yum install -y -q yum-utils'
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
warning: /var/cache/yum/x86_64/7/base/packages/libxml2-python-2.9.1-6.el7.4.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID f4a80
b5: NOKEY
Public key for libxml2-python-2.9.1-6.el7.4.x86_64.rpm is not installed
Public key for yum-utils-1.1.31-54.el7_8.noarch.rpm is not installed
Importing GPG key 0xF4A80EB5:
 Userid     : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"
 Fingerprint: 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5
 Package    : centos-release-7-3.1611.el7.centos.x86_64 (@anaconda)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
+ sh -c 'yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo'
Loaded plugins: fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
+ '[' stable '!=' stable ']'
+ sh -c 'yum makecache'
Loaded plugins: fastestmirror
base                                                                     | 3.6 kB  00:00:00
docker-ce-stable                                                         | 3.5 kB  00:00:00
```

```
Metadata Cache Created
+ '[' -n '' ']'
+ sh -c 'yum install -y -q docker-ce'
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
warning: /var/cache/yum/x86_64/7/docker-ce-stable/packages/containerd.io-1.3.7-3.1.el7.x86_64.rpm: Header V4 RSA/SHA512 Signature, ke
y ID 621e9f35: NOKEY
Public key for containerd.io-1.3.7-3.1.el7.x86_64.rpm is not installed
Importing GPG key 0x621E9F35:
 Userid     : "Docker Release (CE rpm) <docker@docker.com>"
 Fingerprint: 060a 61c5 1b55 8a7f 742b 77aa c52f eb6b 621e 9f35
 From       : https://download.docker.com/linux/centos/gpg
If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

  sudo usermod -aG docker your-user

Remember that you will have to log out and back in for this to take effect!

WARNING: Adding a user to the "docker" group will grant the ability to run
         containers which can be used to obtain root privileges on the
         docker host.
         Refer to https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface
         for more information.
```

（十三）docker 安装方法 4

（1）安装必要的包。

yum install -y yum-utils device-mapper-persistent-data lvm2

```
[root@centos7test ~]#
[root@centos7test ~]# yum install -y yum-utils  device-mapper-persistent-data  lvm2
Loaded plugins: fastestmirror, langpacks
base                                                                    | 3.6 kB  00:00:00
extras                                                                  | 2.9 kB  00:00:00
updates                                                                 | 2.9 kB  00:00:00
updates/7/x86_64/primary_db                                             |  13 MB  00:00:02
Determining fastest mirrors
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
Resolving Dependencies
--> Running transaction check
---> Package device-mapper-persistent-data.x86_64 0:0.6.3-1.el7 will be updated
---> Package device-mapper-persistent-data.x86_64 0:0.8.5-3.el7_9.2 will be an update
---> Package lvm2.x86_64 7:2.02.166-1.el7 will be updated
---> Package lvm2.x86_64 7:2.02.187-6.el7_9.5 will be an update
```

（2）设置 Docker CE 稳定版的仓库地址。

yum-config-manager --add-repo \

http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

```
[root@centos7test ~]# yum-config-manager --add-repo \
>     http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror, langpacks
adding repo from: http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
grabbing file http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
[root@centos7test ~]#
```

查看所有仓库中所有 docker 版本，并选择特定版本安装

（3）# yum list docker-ce --showduplicates | sort -r

```
[root@centos7test ~]#  yum list docker-ce --showduplicates | sort -r
 * updates: mirrors.aliyun.com
Loading mirror speeds from cached hostfile
Loaded plugins: fastestmirror, langpacks
 * extras: mirrors.aliyun.com
docker-ce.x86_64            3:20.10.9-3.el7                     docker-ce-stable
docker-ce.x86_64            3:20.10.8-3.el7                     docker-ce-stable
docker-ce.x86_64            3:20.10.7-3.el7                     docker-ce-stable
docker-ce.x86_64            3:20.10.6-3.el7                     docker-ce-stable
docker-ce.x86_64            3:20.10.5-3.el7                     docker-ce-stable
docker-ce.x86_64            3:20.10.4-3.el7                     docker-ce-stable
docker-ce.x86_64            3:20.10.3-3.el7                     docker-ce-stable
docker-ce.x86_64            3:20.10.2-3.el7                     docker-ce-stable
docker-ce.x86_64            3:20.10.1-3.el7                     docker-ce-stable
docker-ce.x86_64            3:20.10.12-3.el7                    docker-ce-stable
docker-ce.x86_64            3:20.10.11-3.el7                    docker-ce-stable
docker-ce.x86_64            3:20.10.10-3.el7                    docker-ce-stable
docker-ce.x86_64            3:20.10.0-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.9-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.8-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.7-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.6-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.5-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.4-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.3-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.2-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.15-3.el7                    docker-ce-stable
docker-ce.x86_64            3:19.03.14-3.el7                    docker-ce-stable
docker-ce.x86_64            3:19.03.1-3.el7                     docker-ce-stable
docker-ce.x86_64            3:19.03.13-3.el7                    docker-ce-stable
docker-ce.x86_64            3:19.03.12-3.el7                    docker-ce-stable
docker-ce.x86_64            3:19.03.11-3.el7                    docker-ce-stable
docker-ce.x86_64            3:19.03.10-3.el7                    docker-ce-stable
```

（4）安装指定版本的 Docker CE。

yum install -y docker-ce-19.03.2-3.el7 docker-ce-cli-19.03.2-3.el7 containerd.io

```
[root@centos7test ~]# yum install -y docker-ce-20.10.12-3.el7 docker-ce-cli-20.10.12-3.el7 containerd.io
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
Resolving Dependencies
--> Running transaction check
---> Package containerd.io.x86_64 0:1.4.12-3.1.el7 will be installed
--> Processing Dependency: container-selinux >= 2:2.74 for package: containerd.io-1.4.12-3.1.el7.x86_64
---> Package docker-ce.x86_64 3:20.10.12-3.el7 will be installed
--> Processing Dependency: docker-ce-rootless-extras for package: 3:docker-ce-20.10.12-3.el7.x86_64
---> Package docker-ce-cli.x86_64 1:20.10.12-3.el7 will be installed
--> Processing Dependency: docker-scan-plugin(x86-64) for package: 1:docker-ce-cli-20.10.12-3.el7.x86_64
--> Running transaction check
---> Package container-selinux.noarch 2:2.119.2-1.911c772.el7_8 will be installed
--> Processing Dependency: selinux-policy-targeted >= 3.13.1-216.el7 for package: 2:container-selinux-2.119.2-1.911c772.el7_8.noarch
--> Processing Dependency: selinux-policy-base >= 3.13.1-216.el7 for package: 2:container-selinux-2.119.2-1.911c772.el7_8.noarch
--> Processing Dependency: selinux-policy >= 3.13.1-216.el7 for package: 2:container-selinux-2.119.2-1.911c772.el7_8.noarch
--> Processing Dependency: policycoreutils >= 2.5-11 for package: 2:container-selinux-2.119.2-1.911c772.el7_8.noarch
---> Package docker-ce-rootless-extras.x86_64 0:20.10.12-3.el7 will be installed
--> Processing Dependency: fuse-overlayfs >= 0.7 for package: docker-ce-rootless-extras-20.10.12-3.el7.x86_64
--> Processing Dependency: slirp4netns >= 0.4 for package: docker-ce-rootless-extras-20.10.12-3.el7.x86_64
---> Package docker-scan-plugin.x86_64 0:0.12.0-3.el7 will be installed
```

说明：

docker-ce-cli 是用于远程命令管理可以不安装

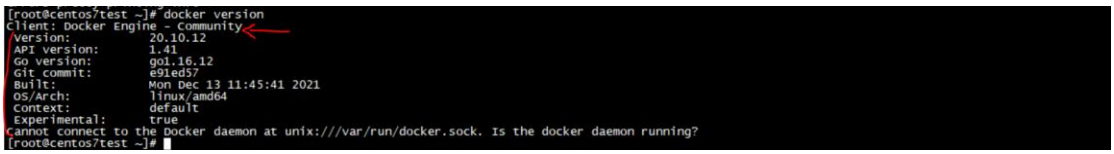
containerd.io 是用于 api 接口的可以不安装

若是安装最新版本

yum install -y docker-ce docker-ce-cli containerd.io

（5）安装完毕，查看版本。

docker version



（6）启动 Docker。

systemctl start docker

（7）运行 hello-world 镜像来验证 Docker CE 已经正常安装。

docker run hello-world

（8）配置 Docker 开机自动启动。

systemctl enable docker

（9）开启 Docker 远程访问。

① 执行 systemctl edit docker 命令打开 docker.service 单元配置文件的 override 文件，添加以下内容。

[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -H unix:///var/run/docker.sock -H tcp://0.0.0.0:2375

② 保存该文件。

③ 执行以下命令重新加载 systemctl 配置。

systemctl daemon-reload

④ 执行以下命令重新启动 Docker。

systemctl restart docker

⑤ 执行以下命令检查确认 Docker 守护进程是否在所配置的端口上侦听。

netstat -lntp | grep dockerd

⑥ 通过-H 选项指定要连接的远程主机。

docker -H tcp://192.168.199.51:2375 info

（十四）docker 安装方法 5（本地离线安装）

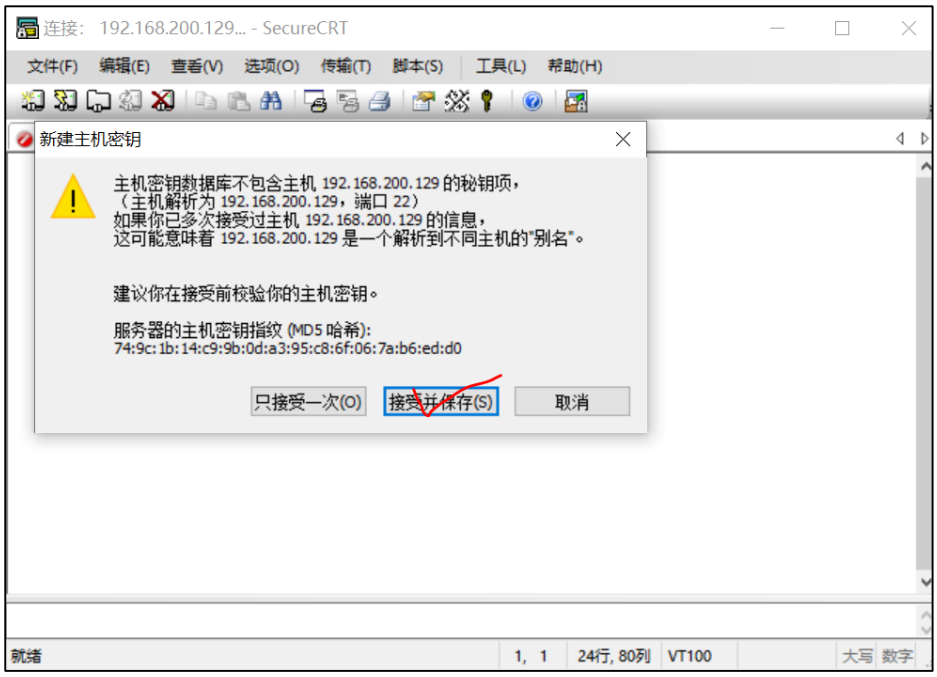
1.安装 centos7.5（1804）以上最小化系统，虚拟机 nat 连接网络

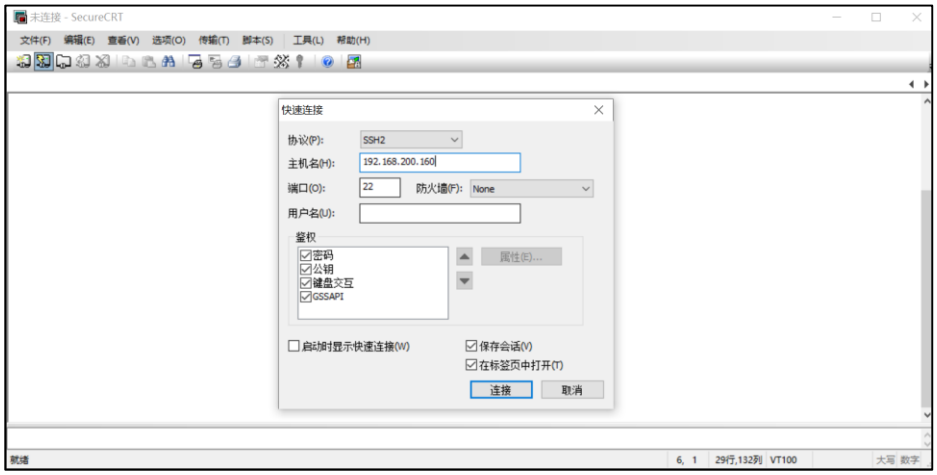
2.开启 ssh

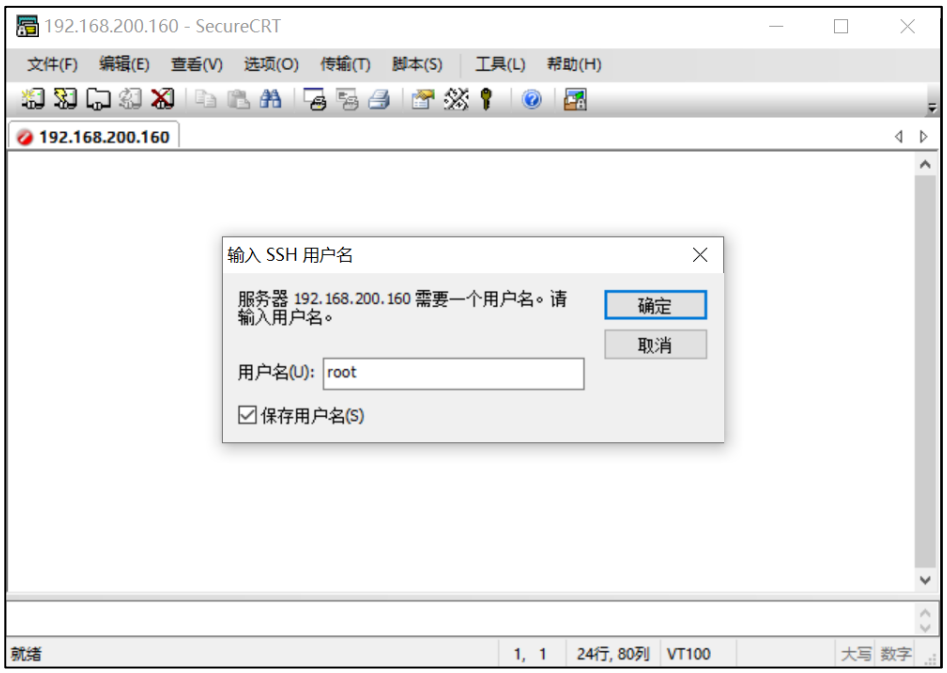
systemctl start sshd

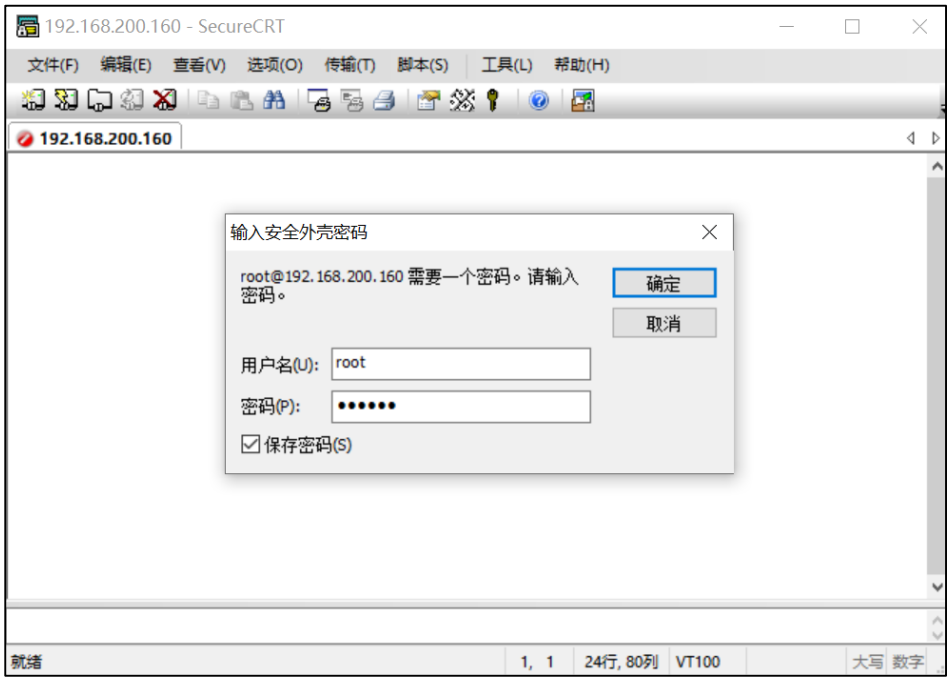
systemctl enable sshd

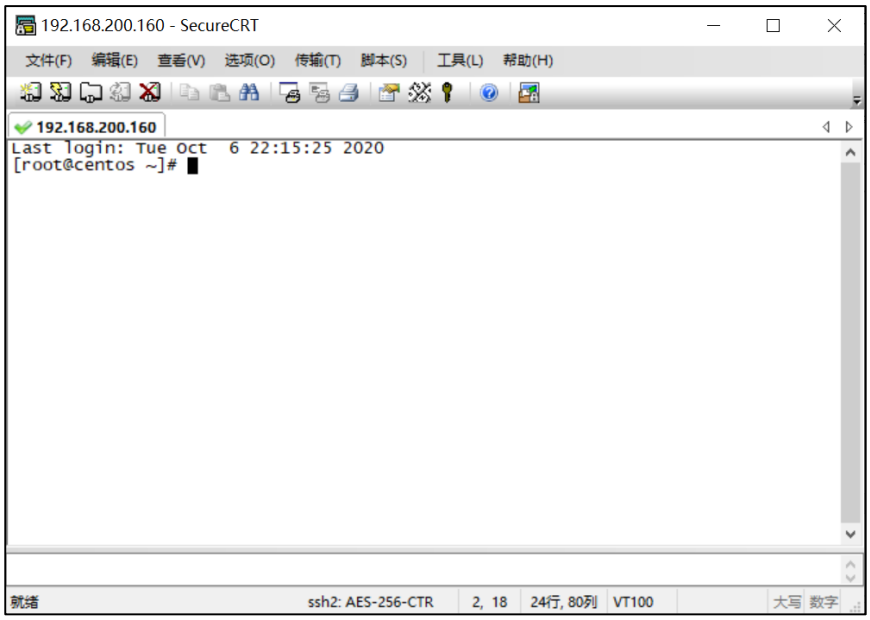
3.ssh 连接

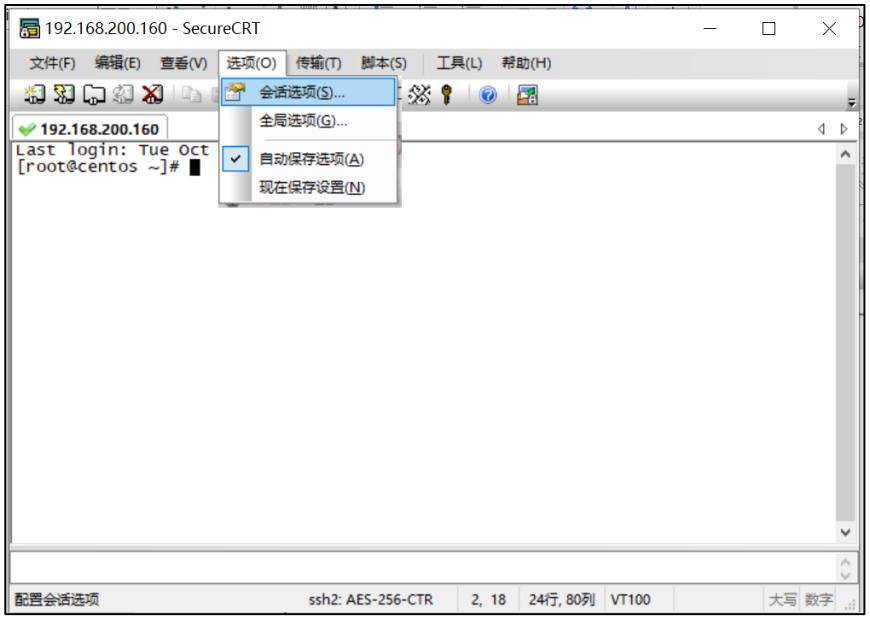


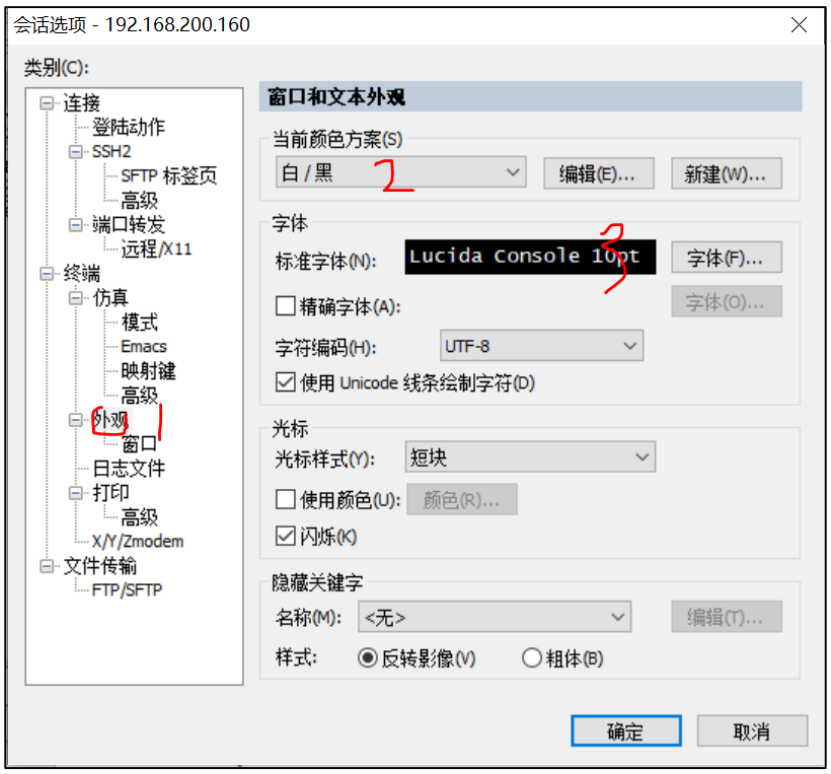


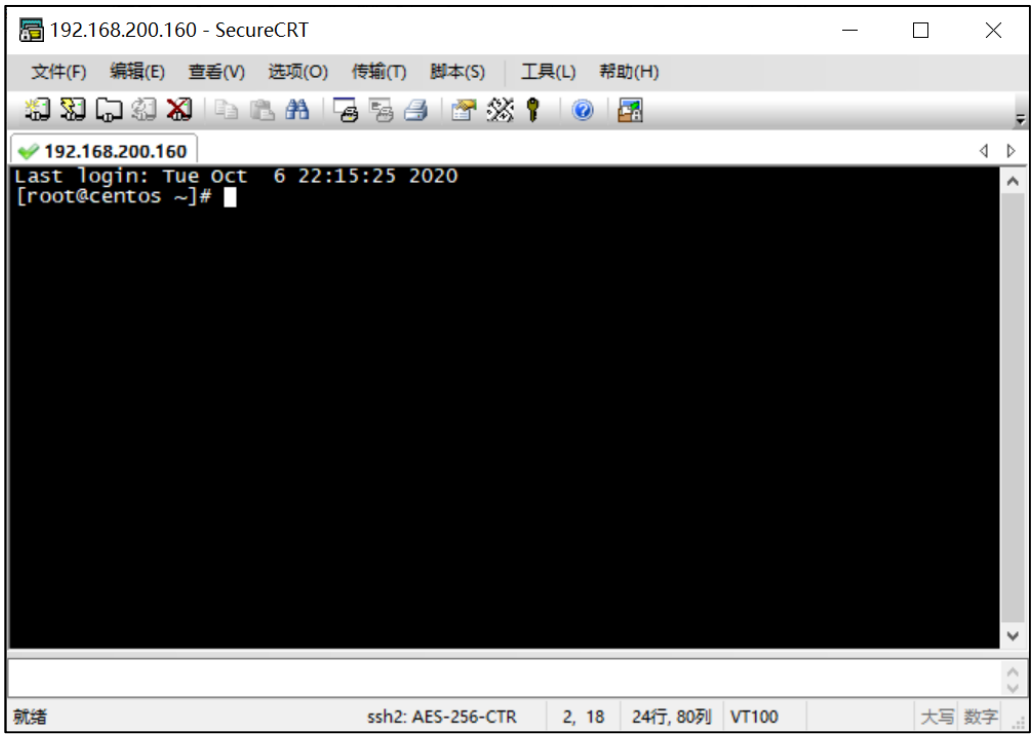

192.168.200.160 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
192.168.200.160
输入 SSH 用户名
服务器 192.168.200.160 需要一个用户名。请输入用户名。
确定
取消
用户名(U): root
保存用户名(S)
就绪
1, 1
24行, 80列
VT100
大写
数字


192.168.200.160 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
192.168.200.160
输入安全外壳密码
root@192.168.200.160 需要一个密码。请输入密码。
确定
取消
用户名(U): root
密码(P): ••••••
保存密码(S)
就绪
1, 1
24行, 80列
VT100
大写
数字


192.168.200.160 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
192.168.200.160
Last login: Tue Oct 6 22:15:25 2020
[root@centos ~]#
就绪
ssh2: AES-256-CTR
2, 18
24行, 80列
VT100
大写 数字


192.168.200.160 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
会话选项(S)...
全局选项(G)...
自动保存选项(A)
现在保存设置(N)
192.168.200.160
Last login: Tue Oct
[root@centos ~]#
配置会话选项
ssh2: AES-256-CTR
2, 18
24行, 80列
VT100
大写 数字





4.关闭防火墙和 selinux

关闭 selinux

（1）永久关闭 selinux

修改/etc/selinux/config 文件

将 SELINUX=enforcing 改为 SELINUX=disabled（**需要 linux 重新启动才能起作用**）

（2）临时关闭 selinux

setenforce 0

关闭防火墙

(1)临时关闭:

Iptables -F

Iptables -X

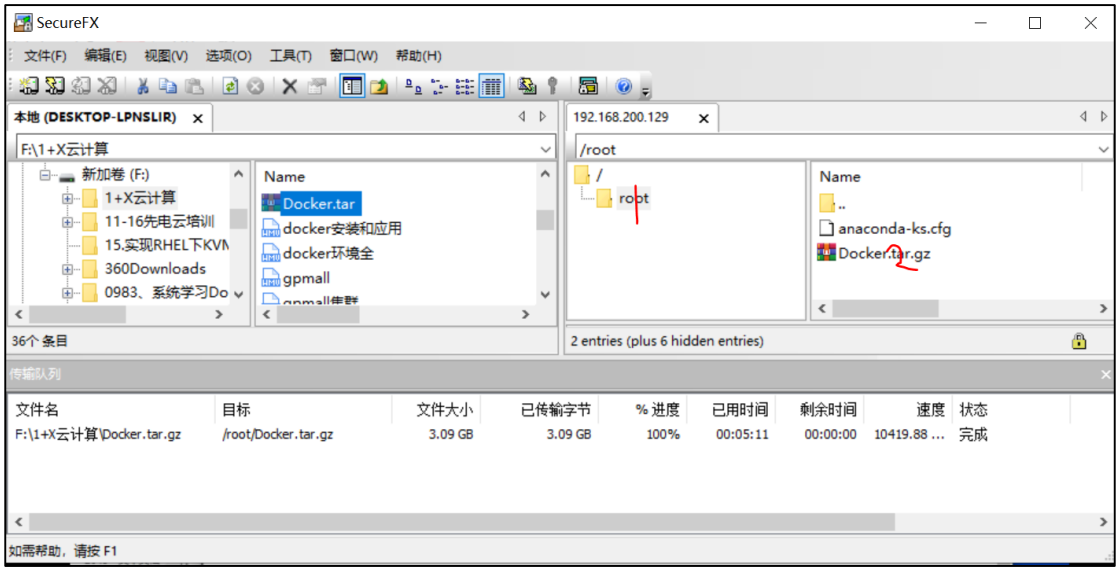
Iptables -Z

(2)永久关闭：

（1）systemctl stop firewalld　停止防火墙

（2）systemctl disable firewalld　开机不启动防火墙

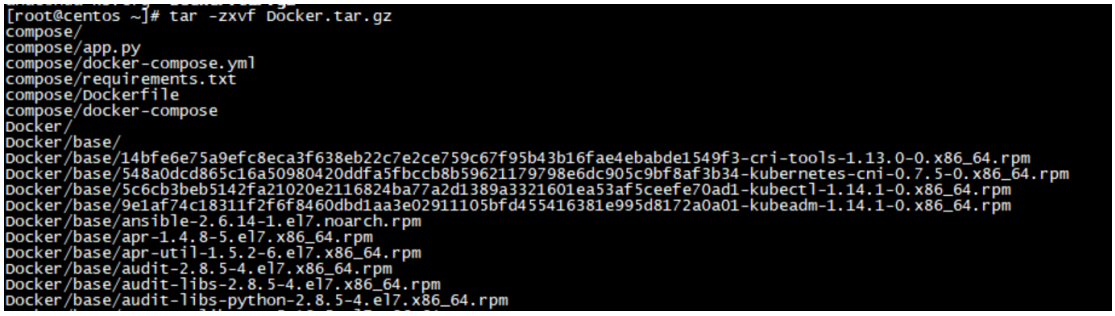
5.上传资料包 Docker.tar.gz 到 linux 的 root 下



6.解压

ls

tar -zxvf Docker.tar.gz



ls

```
[root@centos ~]# ls
anaconda-ks.cfg  compose  Docker  Docker.tar.gz  harbor-offline-installer-v1.5.3.tgz  images
[root@centos ~]#
```

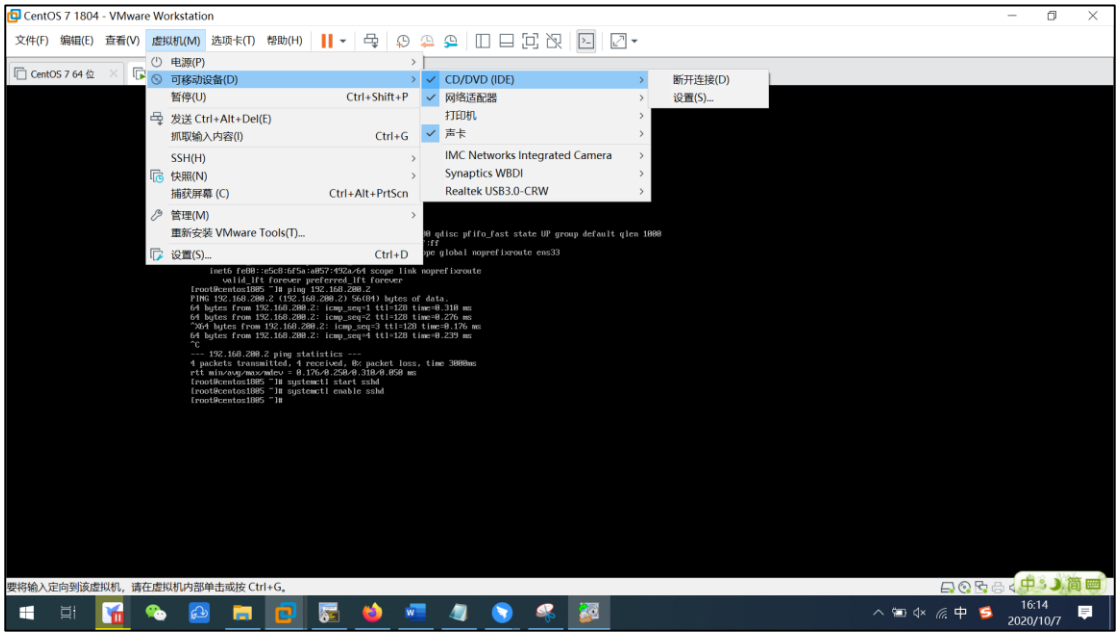
7.设置本地安装源

mkdir /opt/centos

mount /dev/cdrom /opt/centos/

```
[root@centos ~]# mkdir /opt/centos
[root@centos ~]# mount /dev/cdrom /opt/centos/
mount: /dev/sr0 is write-protected, mounting read-only
[root@centos ~]#
```

确保虚拟光盘一定要连接



rm -rf /etc/yum.repos.d/*

```
[root@centos ~]# rm -rf /etc/yum.repos.d/*
[root@centos ~]#
```

vi /etc/yum.repos.d/local.repo

[docker]

name=docker

baseurl=file:///root/Docker

gpgcheck=0

enabled=1

[centos7.5]

name=centos7.5

baseurl=file:///opt/centos

gpgcheck=0

enabled=1

```
[root@centos ~]# vi /etc/yum.repos.d/local.repo

[docker]
name=docker
baseurl=file:///root/Docker
gpgcheck=0
enabled=1
```

yum clean all

清除原有依赖

```
[root@centos ~]# yum clean all
Loaded plugins: fastestmirror
Cleaning repos: docker
Cleaning up everything
Cleaning up list of fastest mirrors
```

yum repolist

生成新的依赖

```
[root@centos1805 ~]# yum repolist
Loaded plugins: fastestmirror
Determining fastest mirrors
centos7.5                                                                | 3.6 kB  00:00:00
docker                                                                   | 2.9 kB  00:00:00
(1/3): centos7.5/group_gz                                                | 166 kB  00:00:00
(2/3): docker/primary_db                                                 | 777 kB  00:00:00
(3/3): centos7.5/primary_db                                              | 3.1 MB  00:00:00
repo id                                  repo name                                status
centos7.5                                centos7.5                                3,971
docker                                   docker                                     341
repolist: 4,312
[root@centos1805 ~]# yum install -y yum-utils device-mapper-persistent-data
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package device-mapper-persistent-data.x86_64 0:0.7.3-3.el7 will be updated
---> Package device-mapper-persistent-data.x86_64 0:0.8.5-1.el7 will be an update
```

8.安装后端存储引擎

yum install -y yum-utils device-mapper-persistent-data

9.安装 docker

#yum install docker-ce

10. 启动 docker

systemctl start docker

随系统启动

systemctl senable docker

11.查看安装的 docker 版本和信息

（1）# docker version

```
[root@centos1805 ~]# docker version
Client:
 Version:           18.09.6
 API version:       1.39
 Go version:        go1.10.8
 Git commit:        481bc77156
 Built:             Sat May  4 02:34:58 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.6
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.8
  Git commit:       481bc77
  Built:            Sat May  4 02:02:43 2019
  OS/Arch:          linux/amd64
  Experimental:     false
[root@centos1805 ~]#
```

（2）# docker info

```
[root@centos1805 ~]# docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 18.09.6
Storage Driver: overlay2
 Backing Filesystem: xfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: b34a5c8af56e510852c35414db4c1f4fa6172339
runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
init version: fec3683
Security Options:
 seccomp
  Profile: default
Kernel Version: 3.10.0-862.el7.x86_64
Operating System: CentOS Linux 7 (Core)
OSType: linux
Architecture: x86_64
CPUs: 2
Total Memory: 1.779GiB
Name: centos1805
ID: SFQE:45EJ:SHOW:NU7J:25D7:JVSL:CZWI:RMGF:F5TP:HZDH:MNFU:6DPT
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
```

二、基础运行容器

（一）安装加速器

docker 镜像在国外网站 docker hub，下载速度慢，通过安装加速器可以加快下载速度

1.安装加速器

curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s http://85d5449d.m.daocloud.io

```
[root@localhost ~]# curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s http://85d5449d.m.daocloud.io
docker version >= 1.12
{"registry-mirrors": ["http://85d5449d.m.daocloud.io"]}
Success.
You need to restart docker to take effect: sudo systemctl restart docker
[root@localhost ~]#
```

或者

curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s http://f1361db2.m.daocloud.io

到加速器官网上查加速器相关，现在国内需要去注册才能获取加速器的地址

常见加速器网址

（1）网易镜像地址：http://hub-mirror.c.163.com

（2）阿里镜像地址：https://dev.aliyun.com/search.html （注册阿里云账号，会出现专属加速器）

（3）DaoCloud 镜像地址：https://www.daocloud.io/ （同上，注册之后会有专属加速器）

2.重新启动 docker 服务生效

systemctl restart docker

```
[root@localhost ~]# systemctl restart docker
[root@localhost ~]# docker run -d -p 8080:80 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
6ec8c9369e08: Pulling fs layer
819d6e0b29e7: Pulling fs layer
6a237d0d4aa4: Pulling fs layer
cd9a987eec32: Waiting
fdec8f3f8485: Waiting
```

（二）运行一个容器

实际上是下载 httpd 镜像，然后运行镜像为容器，把 80（第二个）端口映射为本地 80（第一个）端口；-d 是后台运行，-p 是端口映射

查看是否已经下载镜像

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@localhost ~]#
```

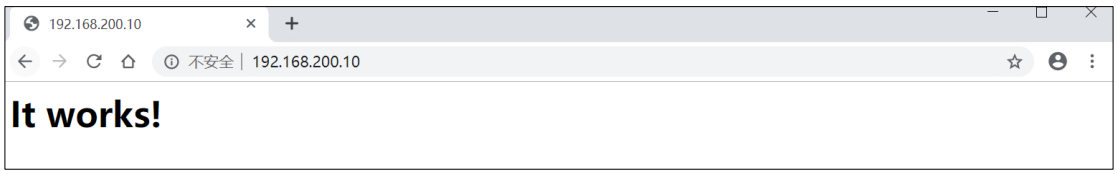
1.下载安装 httpd 镜像

docker run -d -p 80:80 httpd

```
[root@localhost ~]# docker run -d -p 80:80 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
8ec398bc0356: Pulling fs layer
354e6904d655: Pulling fs layer
27298e4c749a: Pulling fs layer
10e27104ba69: Waiting
36412f6b2f6e: Waiting
latest: Pulling from library/httpd
8ec398bc0356: Pull complete
354e6904d655: Pull complete
27298e4c749a: Pull complete
10e27104ba69: Pull complete
36412f6b2f6e: Pull complete
Digest: sha256:769018135ba22d3a7a2b91cb89b8de711562cdf51ad6621b2b9b13e95f3798de
Status: Downloaded newer image for httpd:latest
d5a4eb50a0648026d3af1dc1a1817f2037d8132f3dc62775cdae03103f495eef
[root@localhost ~]#
```

2.真机访问 centos 系统中 docker 服务

http://centos 系统 ip 地址:80



docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              bf756fb1ae65        6 months ago        13.3kB
[root@localhost ~]#
```

（三）查看本地已下载 docker 镜像

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
[root@localhost ~]#
```

（四）查看镜像详细信息

docker inspect hello-world

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              470671670cac        9 days ago          237MB
nginx               latest              c7460dfcab50        2 weeks ago         126MB
httpd               latest              c2aa7e16edd8        4 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        4 weeks ago         1.22MB
registry            latest              f32a97de94e1        10 months ago       25.8MB
hello-world         latest              fce289e99eb9        13 months ago       1.84kB
[root@localhost ~]# docker inspect hello-world
[
    {
        "Id": "sha256:fce289e99eb9bca977dae136fbe2a82b6b7d4c372474c9235adc1741675f587e",
        "RepoTags": [
            "hello-world:latest"
        ],
        "RepoDigests": [
            "hello-world@sha256:9572f7cdcee8591948c2963463447a53466950b3fc15a247fcad1917ca215a2f"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2019-01-01T01:29:27.650294696Z",
        "Container": "8e2caa5a514bb6d8b4f2a2553e9067498d261a0fd83a96aeaaf303943dff6ff9",
        "ContainerConfig": {
            "Hostname": "8e2caa5a514b",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
```

（四）给镜像新命名并加标签

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              470671670cac        9 days ago          237MB
nginx               latest              c7460dfcab50        2 weeks ago         126MB
httpd               latest              c2aa7e16edd8        4 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        4 weeks ago         1.22MB
registry            latest              f32a97de94e1        10 months ago       25.8MB
hello-world         latest              fce289e99eb9        13 months ago       1.84kB
```

docker tag hello-world hello:biaoqian

```
[root@localhost ~]# docker tag hello-world hello:biaoqian
[root@localhost ~]#
```

结果

docker image

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              470671670cac        9 days ago          237MB
nginx               latest              c7460dfcab50        2 weeks ago         126MB
httpd               latest              c2aa7e16edd8        4 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        4 weeks ago         1.22MB
registry            latest              f32a97de94e1        10 months ago       25.8MB
hello-world         latest              fce289e99eb9        13 months ago       1.84kB
hello               biaoqian            fce289e99eb9        13 months ago       1.84kB
[root@localhost ~]#
```

（四）搜索 docker hub 上镜像

搜索 lamp 类镜像

#docker search lamp

```
[root@localhost ~]# docker search lamp
NAME                      DESCRIPTION                                     STARS               OFFICIAL            A
UTOMATED
linode/lamp               LAMP on Ubuntu 14.04.1 LTS Container            179
mattrayner/lamp           A simple LAMP docker image running the prere…   162                                     [
OK]
tutum/lamp                Out-of-the-box LAMP image (PHP+MySQL)           131
greyltc/lamp              a super secure, up-to-date and lightweight L…   99                                      [
OK]
fauria/lamp               Modern, developer friendly LAMP stack. Inclu…   79                                      [
OK]
lioshi/lamp               Docker image for LAMP + MySql under debian      13                                      [
OK]
dgraziotin/lamp                                                           11                                      [
OK]
linuxconfig/lamp          Automated build LAMP stack environment for f…   5                                       [
OK]
zopnow/lamp-stack         Latest Ubuntu LTS with PHP, MySQL, Apache an…   4                                       [
OK]
greyltc/lamp-sshd-aur     LAMP and ssh servers with aur access            4                                       [
OK]
dell/lamp-base            Base LAMP Image for dependant images            3
```

（五）下载 docker hub 上镜像

下载最小的镜像，大小只有 1.8k

docker pull hello-world　//下载镜像

```
[root@localhost ~]# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
1b930d010525: Pulling fs layer
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:9572f7cdcee8591948c2963463447a53466950b3fc15a247fcad1917ca215a2f
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

docker images　　//查看已下载镜像

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
[root@localhost ~]#
```

docker run hello-world　　//运行镜像

```
[root@localhost ~]# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[root@localhost ~]#
```

三、Docker 镜像

设置公有镜像国内源

目的加速镜像下载

1.

vi /etc/docker/daemon.json

{

"registry-mirrors":["https://registry.docker-cn.com"]

}

```
{
"registry-mirrors":["https://registry.docker-cn.com"]
}
~
~
```

2.

systemctl daemon-reload

3.

systemctl restart docker

（一）下载镜像

docker pull centos

```
[root@localhost ~]# docker pull centos
Using default tag: latest
latest: Pulling from library/centos
8a29a15cefae: Pulling fs layer
latest: Pulling from library/centos
8a29a15cefae: Pull complete
Digest: sha256:fe8d824220415eed5477b63addf40fb06c3b049404242b31982106ac204f6700
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest
[root@localhost ~]#
```

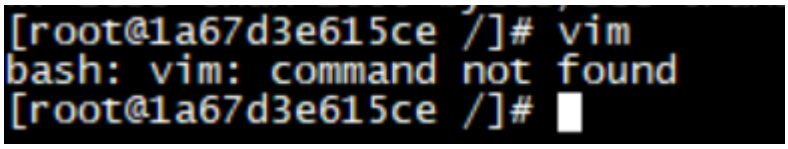
（二）运行镜像

1.运行镜像，以交互式模式进入到容器

docker run -it centos // -it 交互模式

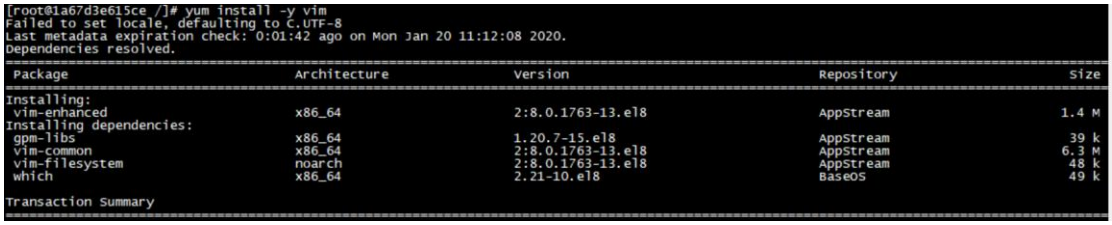
2.给容器安装 vim

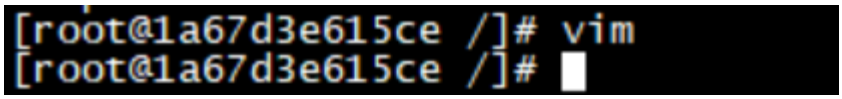
（1）判断 vim 是否安装

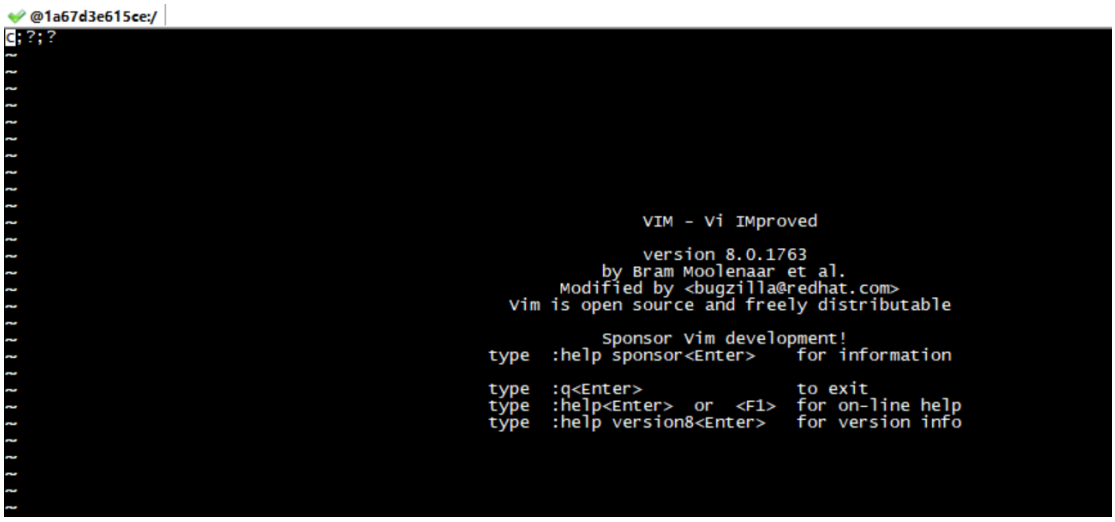


（2）安装 vim

yum install -y vim







3.退出容器

[root@1a67d3e615ce /]# exit

```
[root@1a67d3e615ce /]# exit
exit
[root@localhost ~]#
```

4.查看正在运行的容器

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
a44466a44cdb        centos              "/bin/bash"         13 seconds ago      Up 12 seconds                           cranky_ride
d5a4eb50a064        httpd               "httpd-foreground"  About an hour ago   Up About an hour    0.0.0.0:80->80/tcp  dazzling_gauss
```

docker ps [OPTIONS]

OPTIONS 说明：

- -a ：显示所有的容器，包括未运行的。
- -f ：根据条件过滤显示的内容。
- --format ：指定返回值的模板文件。
- -l ：显示最近创建的容器。
- -n ：列出最近创建的 n 个容器。
- --no-trunc ：不截断输出。
- -q ：静默模式，只显示容器编号。
- -s ：显示总的文件大小。

（三）手工生成新的镜像方法 1（commit）

1.生成新镜像，容器必须在运行

docker run httpd

docker ps -l 显示最新启动的容器

```
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
 PORTS               NAMES
583e4694ef7b        httpd               "httpd-foreground"  About an hour ago   Up 14 seconds
 80/tcp              upbeat_bouman
```

docker commit 583e4994ef7b new-httpd 将容器生成镜像

```
[root@localhost ~]# docker commit 583e4694ef7b new-httpd
sha256:5288058a50b1c162ce406db4183ec5f20ad0d793379eeacda14387d71a893069
```

```
[root@localhost ~]# docker commit cranky_ride new_centos1
sha256:8d2df9544ce0ba7cb8d57f7cc7a6a66ac81024f04435fa33fa2df2d760b9a00f
```

2.查看生成的镜像

docker images //容量改变

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
new-httpd           latest              5288058a50b1        8 seconds ago       165MB
hello-world         latest              bf756fb1ae65        6 months ago        13.3kB
newweb              latest              046c37a21365        8 months ago        126MB
httpd               latest              2ae34abc2ed0        8 months ago        165MB
nginx               latest              231d40e811cd        8 months ago        126MB
[root@localhost ~]#
```

（四）手工生成新的镜像方法 2（commit）

本次操作使用 securecrt 登录操作

1.运行一个镜像

docker run -it --name nginx-man centos

-i 是标准输出

-t 是终端

```
[root@localhost ~]# docker run -it --name nginx-man centos
[root@dca4d436d408 /]#
```

2.安装 wget gcc gcc-c++ make openssl-devel

yum install -y wget gcc gcc-c++ make openssl-devel

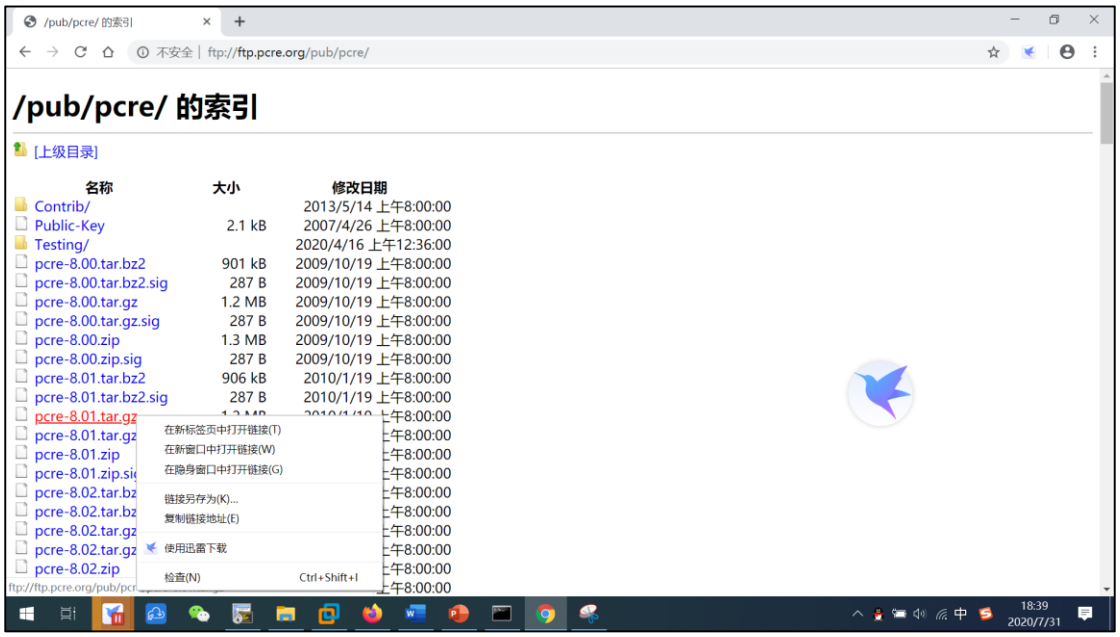
```
[root@dca4d436d408 /]# yum install -y wget gcc gcc-c++ make openssl-devel
Failed to set locale, defaulting to C.UTF-8
CentOS-8 - AppStream                            1.2 MB/s | 5.8 MB
CentOS-8 - Base                                 987 kB/s | 2.2 MB
CentOS-8 - Extras                               5.7 kB/s | 7.0 kB
```

3.获取 nginx 下载地址，在容器内下载 nginx

wget http://nginx.org/download/nginx-1.19.1.tar.gz

```
[root@dca4d436d408 /]# wget http://nginx.org/download/nginx-1.19.1.tar.gz
--2020-07-31 10:20:33--  http://nginx.org/download/nginx-1.19.1.tar.gz
Resolving nginx.org (nginx.org)... 52.58.199.22, 3.125.197.172, 2a05:d014:edb:5702::6, ...
Connecting to nginx.org (nginx.org)|52.58.199.22|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1047223 (1023K) [application/octet-stream]
Saving to: 'nginx-1.19.1.tar.gz'

nginx-1.19.1.tar.gz          62%[=================================>                ] 638.16K   181KB/s    eta 5s
```

4.获取 pcre 下载地址，下载 pcre



wget ftp://ftp.pcre.org/pub/pcre/pcre-8.02.tar.gz

Pcre 是为了安装 nginx 需要

```
[root@dca4d436d408 /]# wget ftp://ftp.pcre.org/pub/pcre/pcre-8.02.tar.gz
--2020-07-31 10:40:39--  ftp://ftp.pcre.org/pub/pcre/pcre-8.02.tar.gz
           => 'pcre-8.02.tar.gz'
Resolving ftp.pcre.org (ftp.pcre.org)... 131.111.8.88
Connecting to ftp.pcre.org (ftp.pcre.org)|131.111.8.88|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD (1) /pub/pcre ... done.
==> SIZE pcre-8.02.tar.gz ... 1247730
==> PASV ... done.    ==> RETR pcre-8.02.tar.gz ... done.
Length: 1247730 (1.2M) (unauthoritative)

pcre-8.02.tar.gz                    3%[=>
```

5.生成 nginx

```
[root@dca4d436d408 /]# ls
bin  etc   lib    lost+found  mnt              opt              proc  run   srv  tmp  var
dev  home  lib64  media       nginx-1.19.1.tar.gz  pcre-8.02.tar.gz  root  sbin  sys  usr
[root@dca4d436d408 /]# pwd
/
[root@dca4d436d408 /]#
```

```
[root@dca4d436d408 /]# mv *.gz /usr/local/src/
[root@dca4d436d408 /]# cd /usr/local/src/
[root@dca4d436d408 src]# ls
nginx-1.19.1.tar.gz  pcre-8.02.tar.gz
[root@dca4d436d408 src]#
```

tar zxf nginx-1.19.1.tar.gz

tar zxf pcre-8.02.tar.gz

```
[root@dca4d436d408 src]# tar zxf nginx-1.19.1.tar.gz
[root@dca4d436d408 src]# tar zxf pcre-8.02.tar.gz
[root@dca4d436d408 src]# ls
nginx-1.19.1  nginx-1.19.1.tar.gz  pcre-8.02  pcre-8.02.tar.gz
[root@dca4d436d408 src]#
```

useradd -s /sbin/nologin -M www

```
[root@dca4d436d408 src]# useradd -s /sbin/nologin -M www
```

cd nginx-1.19.1

```
[root@dca4d436d408 src]# cd nginx-1.19.1
[root@dca4d436d408 nginx-1.19.1]#
```

ls

```
[root@dca4d436d408 nginx-1.19.1]# ls
CHANGES  CHANGES.ru  LICENSE  README  auto  conf  configure  contrib  html  man  src
[root@dca4d436d408 nginx-1.19.1]#
```

./configure --prefix=/usr/local/nginx --user=www --group=www --with-http_ssl_module --with-http_stub_status_module --with-pcre=/usr/local/src/pcre-8.02

```
[root@dca4d436d408 nginx-1.19.1]# ./configure --prefix=/usr/local/nginx --user=www --group=www --with-htt
s_module --with-pcre=/usr/local/src/pcre-8.02
checking for OS
 + Linux 3.10.0-1127.18.2.el7.x86_64 x86_64
checking for C compiler ... found
 + using GNU C compiler
 + gcc version: 8.3.1 20191121 (Red Hat 8.3.1-5) (GCC)
checking for gcc -pipe switch ... found
checking for -Wl,-E switch ... found
checking for gcc builtin atomic operations ... found
checking for C99 variadic macros ... found
checking for gcc variadic macros ... found
checking for gcc builtin 64 bit byteswap ... found
checking for unistd.h ... found
```

make

```
[root@dca4d436d408 nginx-1.19.1]# make
```

```
        src/os/unix/ngx_udp_sendmsg_chain.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g  -I src/core -I src/even
c/event/modules -I src/os/unix -I /usr/local/src/pcre-8.02 -I objs \
        -o objs/src/os/unix/ngx_channel.o \
        src/os/unix/ngx_channel.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g  -I src/core -I src/even
c/event/modules -I src/os/unix -I /usr/local/src/pcre-8.02 -I objs \
        -o objs/src/os/unix/ngx_shmem.o \
        src/os/unix/ngx_shmem.c
cc -c -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -g  -I src/core -I src/even
c/event/modules -I src/os/unix -I /usr/local/src/pcre-8.02 -I objs \
        -o objs/src/os/unix/ngx_process.o \
        src/os/unix/ngx_process.c
```

```
-ldl -lpthread -lcrypt /usr/local/src/pcre-8.02/.libs/libpcre.a -lssl -lcrypto -ldl -lpthread -lz \
-Wl,-E
sed -e "s|%%PREFIX%%|/usr/local/nginx|" \
        -e "s|%%PID_PATH%%|/usr/local/nginx/logs/nginx.pid|" \
        -e "s|%%CONF_PATH%%|/usr/local/nginx/conf/nginx.conf|" \
        -e "s|%%ERROR_LOG_PATH%%|/usr/local/nginx/logs/error.log|" \
        < man/nginx.8 > objs/nginx.8
make[1]: Leaving directory '/usr/local/src/nginx-1.19.1'
[root@dca4d436d408 nginx-1.19.1]#
```

make install

```
[root@dca4d436d408 nginx-1.19.1]# make install
make -f objs/Makefile install
make[1]: Entering directory '/usr/local/src/nginx-1.19.1'
test -d '/usr/local/nginx' || mkdir -p '/usr/local/nginx'
test -d '/usr/local/nginx/sbin' \
        || mkdir -p '/usr/local/nginx/sbin'
test ! -f '/usr/local/nginx/sbin/nginx' \
        || mv '/usr/local/nginx/sbin/nginx' \
              '/usr/local/nginx/sbin/nginx.old'
cp objs/nginx '/usr/local/nginx/sbin/nginx'
test -d '/usr/local/nginx/conf' \
```

6.编辑网站发布目录

vi /usr/local/nginx/conf/nginx.conf

```
daemon off;
#user  nobody;
worker_processes  1;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid        logs/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       mime.types;
    default_type  application/octet-stream;
```

7.exit 退出容器

```
[root@dca4d436d408 nginx-1.19.1]# exit
exit
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
   PORTS               NAMES
dca4d436d408        centos              "/bin/bash"         About an hour ago   Exited (0) 11 seconds ag
                    nginx-man
[root@localhost ~]#
```

8.查看容器进程

docker ps -l

```
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE                  COMMAND             CREATED             STATUS                      PORTS
          NAMES
8d5143ae15c5        changan/my-nginx:v1    "/bin/bash"         2 minutes ago       Exited (0) 12 seconds ago
          stupefied_shtern
```

9.生成镜像

docker commit -m "banben" 8d5143ae15c5 changan/my-nginx:v

```
[root@localhost ~]# docker commit -m "banben"   8d5143ae15c5 changan/my-nginx:v
sha256:9832ccbba7a56b715831054b07e81280aec6a8a3e54672a2ed6463caf294f626
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
changan/my-nginx    v                   9832ccbba7a5        31 seconds ago      444MB
changan/my-nginx    v1                  8c58bf9bcf90        25 minutes ago      444MB
busybox             latest              018c9d7b792b        3 days ago          1.22MB
httpd               latest              9d2a0c6e5b57        8 days ago          166MB
nginx               latest              8cf1bfb43ff5        9 days ago          132MB
centos              latest              831691599b88        6 weeks ago         215MB
hello-world         latest              bf756fb1ae65        7 months ago        13.3kB
```

10.将生成的镜像运行

docker run -d -p 8090:80 changan/my-nginx:v /usr/local/nginx/sbin/nginx

必须加 v，docker 默认加标记 latest

```
[root@localhost ~]# docker run -d -p 8090:80 changan/my-nginx:v /usr/local/nginx/sbin/nginx
ebfa243bef4a91bdbbc8baebe1bdb48f10cf677ccc1888a3a8d02baa0f53cc03
```

11.查看容器运行效果

```
[root@localhost ~]# curl http://127.0.0.1:8090
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@localhost ~]#
```

12.访问服务

http://192.216.5.109:8090

192.216.5.109 是宿主机 IP 地址

（五）Dockerfile 生成新的镜像 1（centos）

1. 建立 Dockerfile 文件

vi Dockerfile

```
[root@centos7test test]# vi Dockerfile

ADD hello /
FROM centos
CMD echo "this is a test"
~
```

2. 生成镜像

docker build --tag centos1 .

```
[root@centos7test test]# docker build --tag centos1 .
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM centos
 ---> 5d0da3dc9764
Step 2/2 : CMD echo "this is a test"
 ---> Running in a1f79905b8fe
Removing intermediate container a1f79905b8fe
 ---> c76a60948bf1
Successfully built c76a60948bf1
Successfully tagged centos1:latest
```

3.运行镜像

docker run centos1

```
[root@centos7test test]# docker run centos1
this is a test
[root@centos7test test]#
```

（五）Dockerfile 生成新的镜像 2（centos）

1.制作可执行脚本文件

（1）# vi hello

```
[root@centos7test test]# vi hello

#!/bin/sh
echo 'hello word'
```

（2）赋予执行权限

chmod a+x hello

2. 建立 Dockerfile 文件

vi Dockerfile

```
[root@centos7test test]# vi Dockerfile

FROM centos
ADD hello /
CMD ["/hello"]
```

3.生成镜像

docker build --tag centostest .

```
[root@centos7test test]# docker build --tag centostest .
Sending build context to Docker daemon  3.072kB
Step 1/3 : FROM centos
latest: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
 ---> 5d0da3dc9764
Step 2/3 : ADD hello /
 ---> 43f910e5c8e0
Step 3/3 : CMD ["/hello"]
 ---> Running in 697c2c05f49c
Removing intermediate container 697c2c05f49c
 ---> 66af13acf010
Successfully built 66af13acf010
Successfully tagged centostest:latest
```

4.执行镜像

docker run centostest

```
[root@centos7test test]# docker run centostest
hello word
[root@centos7test test]#
```

（五）Dockerfile 生成新的镜像 1（centos-nginx）新

1.建立 Dockerfile 文件

```
[root@centos7test ~]# vi Dockerfile

FROM centos:7
Run /usr/bin/yum install -y wget proc-devel net-tools gcc zlib zlib-devel make openssl-devel
RUN wget http://nginx.org/download/nginx-1.9.7.tar.gz
RUN tar zxf nginx-1.9.7.tar.gz
WORKDIR nginx-1.9.7
RUN ./configure --prefix=/usr/local/nginx
RUN make
RUN make install
EXPOSE 80
EXPOSE 443
RUN echo "daemon off;">>/usr/local/nginx/conf/nginx.conf
WORKDIR /root/nginx
ADD run.sh /run.sh
RUN chmod 775 /run.sh
CMD ["/run.sh"]
```

FROM centos:7

Run /usr/bin/yum install -y wget proc-devel net-tools gcc zlib zlib-devel make openssl-devel

RUN wget http://nginx.org/download/nginx-1.9.7.tar.gz

RUN tar zxf nginx-1.9.7.tar.gz

WORKDIR nginx-1.9.7

RUN ./configure --prefix=/usr/local/nginx

RUN make

RUN make install

EXPOSE 80

EXPOSE 443

RUN echo "daemon off;">>/usr/local/nginx/conf/nginx.conf

WORKDIR /root/nginx

ADD run.sh /run.sh

RUN chmod 775 /run.sh

CMD ["/run.sh"]

2.建立脚本

vi run.sh

```
[root@centos7test ~]# vi run.sh

#!/bin/bash
/usr/local/nginx/sbin/nginx
```

3.运行镜像

docker run -d -P nginxtest　大写 p

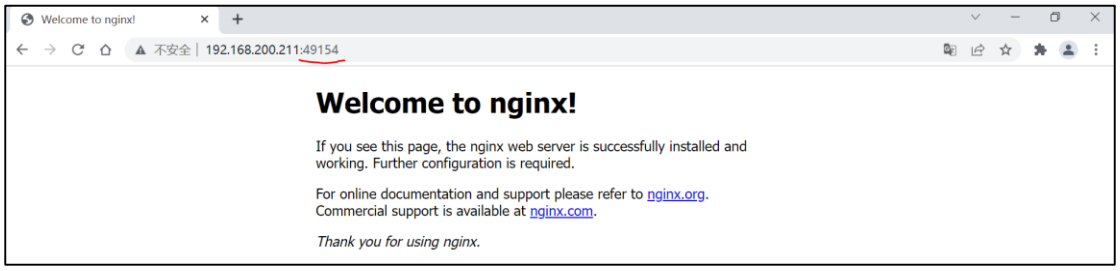
```
[root@centos7test ~]# docker run -d -P nginxtest
5a3aea2594decdd96d43e9abb8588e57fae25f6beff8375309cb8f0d5c245c0c
```

4.查看容器进程

docker ps -l

```
[root@centos7test ~]# docker ps -l
CONTAINER ID   IMAGE       COMMAND     CREATED          STATUS          PORTS                                                                                  NAMES
5a3aea2594de   nginxtest   "/run.sh"   24 seconds ago   Up 23 seconds   0.0.0.0:49154->80/tcp, :::49154->80/tcp, 0.0.0.0:49153->443/tcp, :::49153->443/tcp   wonderful_gould
[root@centos7test ~]#
```

5.访问



（五）Dockerfile 生成新的镜像 2（centos-tomcat）新

1.建立 Dockerfile 文件

vi Dockerfile

```
[root@centos7test ~]# vi Dockerfile
FROM centos:7
ADD jdk-8u181-linux-x64.tar.gz /usr/local
RUN mv /usr/local/jdk1.8.0_181  /usr/local/jdk8u181
ENV JAVA_HOME /usr/local/jdk8u181
ENV JAVA_BIN /usr/local/jdk8u181/bin
ENV JRE_HOME /usr/local/jdk8u181/jre
ENV PATH $PATH:/usr/local/jdk8u181/bin:/usr/local/jdk8u181/jre
ENV CLASSPATH /usr/local/jdk8u181/jre/bin:usr/local/jdk8u181/lib:usr/local/jdk8u181/jre/lib/charsets.jar
ADD apache-tomcat-8.5.75.tar.gz  /usr/local
RUN mv  /usr/local/apache-tomcat-8.5.75  /usr/local/tomcat
EXPOSE 8080
~
```

FROM centos:7

ADD jdk-8u181-linux-x64.tar.gz /usr/local

RUN mv /usr/local/jdk1.8.0_181 /usr/local/jdk8u181

ENV JAVA_HOME /usr/local/jdk8u181

ENV JAVA_BIN /usr/local/jdk8u181/bin

ENV JRE_HOME /usr/local/jdk8u181/jre

ENV PATH $PATH:/usr/local/jdk8u181/bin:/usr/local/jdk8u181/jre

ENV CLASSPATH
/usr/local/jdk8u181/jre/bin:usr/local/jdk8u181/lib:usr/local/jdk8u181/jre/lib/charsets.jar

ADD apache-tomcat-8.5.75.tar.gz /usr/local

RUN mv /usr/local/apache-tomcat-8.5.75 /usr/local/tomcat

EXPOSE 8080

2.生成镜像

docker build -t tomcat6 .

3.镜像运行

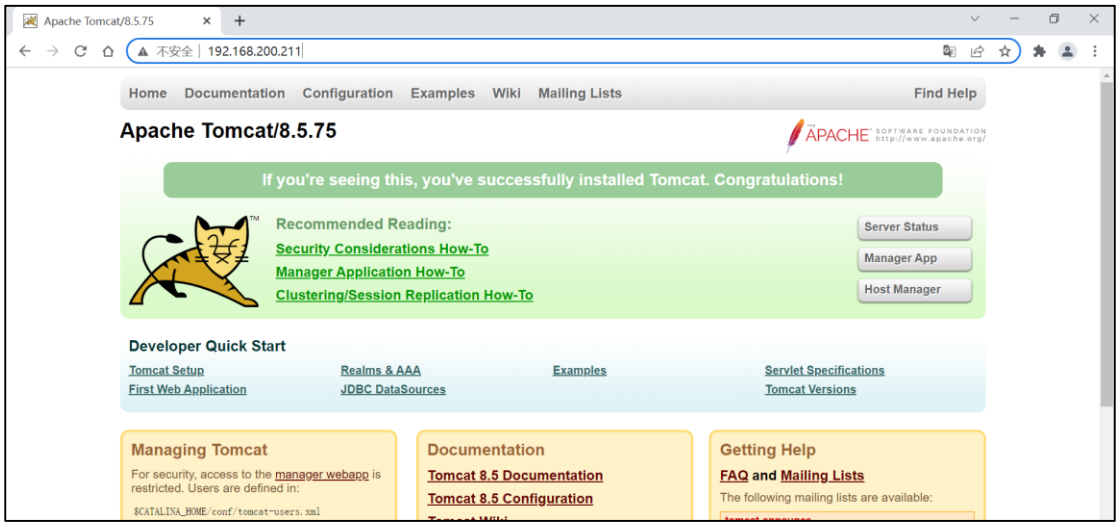
docker run -it -p 80:8080 tomcat6

4.进入容器启动 tomcat

/usr/local/tomcat/bin/startup.sh

```
[root@199bb88f7b6a /]# /usr/local/tomcat/bin/startup.sh
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /usr/local/jdk8u181/jre
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
[root@199bb88f7b6a /]#
```

5.客户端访问



（五）Dockerfile 生成新的镜像 3（centos-mariadb）新

1.建立 Dockerfile 文件

```
# vi Dockerfile
#使用的基础镜像
FROM centos:7
#添加作者信息
MAINTAINER liuxin 842887233@qq.com
#安装 mariadb 数据库
RUN yum -y install mariadb-server
#设置环境变量，便于管理
ENV MARIADB_USER root
ENV MARIADB_PASS maya.123
#让容器支持中文
ENV LC_ALL en_US.UTF-8
#初始化数据库
ADD db_init.sh /root/db_init.sh
RUN chmod 775 /root/db_init.sh
RUN /root/db_init.sh
#导出端口
EXPOSE 3306
#添加启动文件
ADD run.sh /root/run.sh
RUN chmod 775 /root/run.sh
```

#设置默认启动命令

CMD ["/root/run.sh"]

```
[root@centos7test mariadb]# vi Dockerfile

#使用的基础镜像
FROM centos:7

#添加作者信息
MAINTAINER liuxin 842887233@qq.com

#安装mariadb数据库
RUN yum -y install mariadb-server

#设置环境变量，便于管理
ENV MARIADB_USER root
ENV MARIADB_PASS maya.123
#让容器支持中文
ENV LC_ALL en_US.UTF-8

#初始化数据库
ADD db_init.sh /root/db_init.sh
RUN chmod 775 /root/db_init.sh
RUN /root/db_init.sh

#导出端口
EXPOSE 3306

#添加启动文件
ADD run.sh /root/run.sh
RUN chmod 775 /root/run.sh

#设置默认启动命令
CMD ["/root/run.sh"]
```

2.编辑数据初始化脚本

#!/bin/bash

启动

mysql_install_db --user=mysql

sleep 3

mysqld_safe &

sleep 3

mysqladmin -u "$MARIADB_USER" password "$MARIADB_PASS"

授权

mysql -u "$MARIADB_USER" -p"$MARIADB_PASS" -e "use mysql; grant all privileges on *.* to '$MARIADB_USER'@'%' identified by '$MARIADB_PASS' with grant option;"

h=$(hostname)

设置用户密码

mysql -u "$MARIADB_USER" -p"$MARIADB_PASS" -e "use mysql; update user set password=password('$MARIADB_PASS') where user='$M ARIADB_USER' and host='$h';"

创建默认数据库

#mysql -u "$MARIADB_USER" -p"$MARIADB_PASS" -e "CREATE DATABASE IF NOT EXISTS default_db DEFAULT CHARSET utf8 COLLATE utf8_general_ci;"

刷新权限

mysql -u "$MARIADB_USER" -p"$MARIADB_PASS" -e "flush privileges;"

```
[root@centos7test mariadb]# vi db_init.sh

#!/bin/bash

# 启动
mysql_install_db --user=mysql
sleep 3
mysqld_safe &
sleep 3
mysqladmin -u "$MARIADB_USER" password "$MARIADB_PASS"

# 授权
mysql -u "$MARIADB_USER" -p"$MARIADB_PASS" -e "use mysql; grant all privileges on *.* to '$MARIADB_USER'@'%' identified by
'$MARIADB_PASS' with grant option;"

h=$(hostname)

# 设置用户密码
mysql -u "$MARIADB_USER" -p"$MARIADB_PASS" -e "use mysql; update user set password=password('$MARIADB_PASS') where user='$M
ARIADB_USER' and host='$h';"

# 创建默认数据库
#mysql -u "$MARIADB_USER" -p"$MARIADB_PASS" -e "CREATE DATABASE IF NOT EXISTS default_db DEFAULT CHARSET utf8 COLLATE utf8_general_ci;"

# 刷新权限
mysql -u "$MARIADB_USER" -p"$MARIADB_PASS" -e "flush privileges;"
~
```

3.编辑运行脚本

vi run.sh

#!/bin/bash

mysqld_safe

4.生成镜像

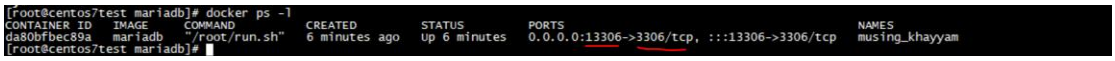
docker build -t mariadb .

5.运行镜像

docker run -d -p 13306:3306 mariadb /root/run.sh

6.查看进程

docker ps -l



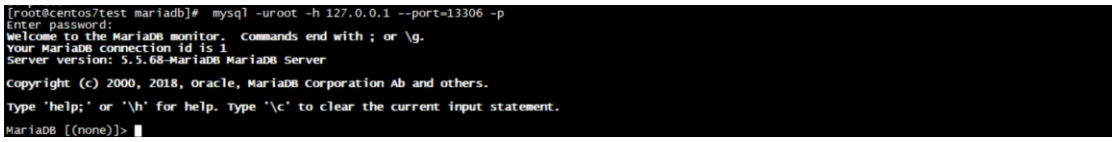
7.下载 mariadb 客户端

yum install -y mariadb

8.登陆验证

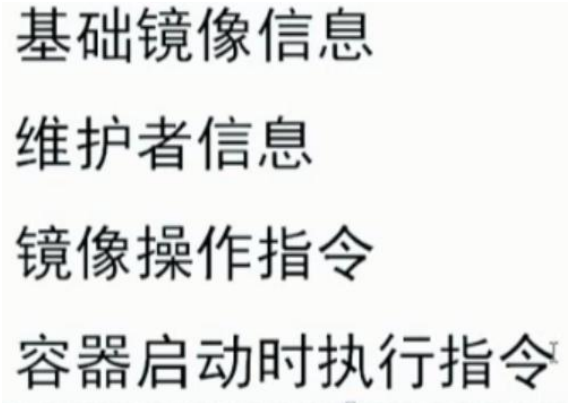
mysql -uroot -h 127.0.0.1 --port=13306 -p

密码在 dockerfile 中设定，为 maya.123



（五）Dockerfile 生成新的镜像 1（centos-nginx）

使用 Dockerfile 生成新的镜像一般方法



1.建立工作目录

mkdir /opt/docker-file

cd /opt/docker-file/

```
[root@localhost ~]# mkdir /opt/docker-file
[root@localhost ~]# cd /opt/docker-file/
[root@localhost docker-file]#
```

mkdir nginx

cd nginx/

```
[root@localhost docker-file]# mkdir nginx
[root@localhost docker-file]# cd nginx/
[root@localhost nginx]# pwd
/opt/docker-file/nginx
[root@localhost nginx]#
```

2.上传相关文件到/opt/nginx 下载

```
[root@centos7test opt]# ls
containerd  docker-file  Dockerfile  nginx-1.19.1.tar.gz  node-bulletin-board  pcre-8.02.tar.gz  rh
[root@centos7test opt]#
```

下面是直接下载，但现在 pcre 在这个网址下载不了了

（https://sourceforge.net/projects/pcre/files/pcre/8.02/），nginx 还可以下载

(# wget ftp://ftp.pcre.org/pub/pcre/pcre-8.02.tar.gz

```
[root@localhost ~]# cd /opt/docker-file/nginx/
[root@localhost nginx]#  wget ftp://ftp.pcre.org/pub/pcre/pcre-8.02.tar.gz
--2020-08-01 09:07:57--  ftp://ftp.pcre.org/pub/pcre/pcre-8.02.tar.gz
           => 鈥榩cre-8.02.tar.gz鈥 Resolving ftp.pcre.org (ftp.pcre.org)... 1
.111.8.88
Connecting to ftp.pcre.org (ftp.pcre.org)|131.111.8.88|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD (1) /pub/pcre ... done.
==> SIZE pcre-8.02.tar.gz ... 1247730
==> PASV ... done.    ==> RETR pcre-8.02.tar.gz ... done.
Length: 1247730 (1.2M) (unauthoritative)

69% [=========================>             ] 863,564      238KB/s  eta 2s
```

wget http://nginx.org/download/nginx-1.19.1.tar.gz

```
[root@localhost nginx]# wget http://nginx.org/download/nginx-1.19.1.tar.gz
--2020-08-01 09:08:43--  http://nginx.org/download/nginx-1.19.1.tar.gz
Resolving nginx.org (nginx.org)... 3.125.197.172, 52.58.199.22
Connecting to nginx.org (nginx.org)|3.125.197.172|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1047223 (1023K) [application/octet-stream]
Saving to: 鈥榥ginx-1.19.1.tar.gz鈥
25% [=========>                              ] 269,285     98.9KB/s
```

)

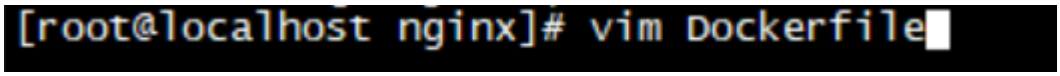
ls

```
[root@localhost nginx]# ls
nginx-1.19.1.tar.gz  pcre-8.02.tar.gz
[root@localhost nginx]#
```

3.Dockerfile 文件编辑

- FROM 他的妈妈是谁（基础镜像）
- MAITAINER 告诉别人，你创造了他（创建维护者信息）
- RUN 你想让他干什么（把命令前面加上 RUN）
- ADD 往他肚子里放点文件（就是往镜像里 COPY 文件，ADD 会自动解压包）
- WORKDIR 我是 cd，今天刚化了妆（当前工作目录）
- VOLUME 给我一个厨房行李的地方（目录挂载）
- EXPOSE 我要打开的门是啥（开放的端口）
- CMD 奔跑吧，兄弟（进程要一直运行下去）

#vi Dockerfile




```
#this is my first Dockerfile
#version 1.0
#author changan

#Base images
FROM centos

#MAINTAINER
MAINTAINER wangchangan

#DockerfileADD
ADD pcre-8.02.tar.gz  /usr/local/src
ADD nginx-1.19.1.tar.gz  /usr/local/src

#RUN
RUN yum install -y wget gcc gcc-c++ make openssl-devel
RUN useradd -s /sbin/nologin  -M www

#WORKDIR
WORKDIR /usr/local/src/nginx-1.19.1

RUN ./configure --prefix=/usr/local/nginx --user=www --group=www --with-http_ssl
_module --with-http_stub_status_module --with-pcre=/usr/local/src/pcre-8.02 && m
ake && make install

RUN echo "daemonoff;"  >>/usr/local/nginx/conf/nginx.conf

ENV PATH /usr/local/nginx/sbin:$PATH
EXPOSE 80

CMD ["nginx"]
```

#this is my first Dockerfile

#version 1.0

#author changan

#Base images

FROM centos:7

#MAINTAINER

MAINTAINER wangchangan

#DockerfileADD

ADD pcre-8.02.tar.gz /usr/local/src

ADD nginx-1.19.1.tar.gz /usr/local/src

#RUN

RUN yum install -y wget gcc gcc-c++ make openssl-devel

RUN useradd -s /sbin/nologin -M www

#WORKDIR

WORKDIR /usr/local/src/nginx-1.19.1

RUN ./configure --prefix=/usr/local/nginx --user=www --group=www --with-http_ssl_module --with-http_stub_status_module --with-pcre=/usr/local/src/pcre-8.02 && make && make install

RUN echo "daemon off;" >>/usr/local/nginx/conf/nginx.conf

ENV PATH /usr/local/nginx/sbin:$PATH

EXPOSE 80

CMD {"nginx"}

4.生成镜像

docker build -t nginx-file:v1 /opt/docker-file/nginx

```
[root@localhost nginx]# docker build -t nginx-file:v1 /opt/docker-file/nginx
Sending build context to Docker daemon  2.299MB
Step 1/12 : FROM centos
 ---> 831691599b88
Step 2/12 : MAINTAINER wangchangan
 ---> Running in 2679f3c2753b
Removing intermediate container 2679f3c2753b
 ---> c59ce11c677a
Step 3/12 : ADD pcre-8.02.tar.gz  /usr/local/src
 ---> 9c0bb9db84ae
Step 4/12 : ADD nginx-1.19.1.tar.gz  /usr/local/src
 ---> e79dbc0a4427
Step 5/12 : RUN yum install -y wget gcc gcc-c++ make openssl-devel
 ---> Running in af7c7acb72b4
```

docker images

```
[root@localhost nginx]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx-file          v1                  c456f2e281ea        About a minute ago  436MB
busybox             latest              018c9d7b792b        4 days ago          1.22MB
httpd               latest              9d2a0c6e5b57        9 days ago          166MB
nginx               latest              8cf1bfb43ff5        9 days ago          132MB
centos              latest              831691599b88        6 weeks ago         215MB
hello-world         latest              bf756fb1ae65        7 months ago        13.3kB
[root@localhost nginx]#
```

5.运行镜像测试

docker run -d -p 80 nginx-file:v1 nginx

注意运行一定要加上标记 v1

```
[root@master ~]# docker run -d -p 80 nginx-file:v1 nginx
c021f4b7fa8934c0a3c02a7fce722a78ea0966254ce3e00d278fc9c11b99536e
```

docker ps

```
[root@master ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
              PORTS                   NAMES
c021f4b7fa89        nginx-file:v1       "nginx"             28 seconds ago      Up 27
seconds       0.0.0.0:32768->80/tcp   sharp_wu
[root@master ~]#
```

http://192.168.100.220:32768

192.168.100.220 是宿主机地址

32768 是映射端口

（五）Dockerfile 生成新的镜像 1（centos-nginx）

1.创建一个工作目录

mkdir dockerfile

```
[root@localhost /]# cd /opt
[root@localhost opt]# ls
containerd  rh
[root@localhost opt]# mkdir dockerfile
[root@localhost opt]# cd dockerfile/
[root@localhost dockerfile]#
```

2.创建 Dockerfile 文件

vim Dockerfile

必须是这个名字，区分大小写

```
[root@centos7test test]# vi Dockerfile

FROM centos:centos7
MAINTAINER CHANGAN "123@QQ.COM"
RUN yum install -y wget
RUN yum install -y epel-release
RUN yum install -y nginx
EXPOSE 80
```

FROM centos:7

MAINTAINER CHANGAN "123@QQ.COM"

RUN yum install -y epel-release

RUN yum install -y nginx

EXPOSE 80

3. 生成镜像文件

docker build -t cnetos-nginx .

注意最后那个点，点表示 Dockerfile 文件在当前目录

```
[root@localhost dockerfile]# docker build -t cnetos-nginx .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM centos
 ---> 831691599b88
Step 2/4 : MAINTAINER CHANGAN "123@QQ.COM"
 ---> Running in 81fdf55808f4
Removing intermediate container 81fdf55808f4
 ---> 027b13b1f7bf
Step 3/4 : RUN yum install -y nginx
 ---> Running in 1809be9efbeb
CentOS-8 - AppStream                            639 kB/s | 5.8 MB     00:09
CentOS-8 - Base                                 379 kB/s | 2.2 MB     00:06
CentOS-8 - Extras                               2.0 kB/s | 7.0 kB     00:03
Dependencies resolved.
================================================================================
 Package                    Arch    Version                              Repo
================================================================================
```

4. 查看生成的镜像

docker images

```
[root@localhost dockerfile]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
cnetos-nginx        latest              51ae5c37de3f        7 seconds ago       290MB
busybox             latest              018c9d7b792b        5 days ago          1.22MB
httpd               latest              9d2a0c6e5b57        10 days ago         166MB
nginx               latest              8cf1bfb43ff5        11 days ago         132MB
centos              latest              831691599b88        6 weeks ago         215MB
hello-world         latest              bf756fb1ae65        7 months ago        13.3kB
[root@localhost dockerfile]#
```

5. 运行生成的镜像

docker run -d -p 80 cnetos-nginx nginx -g "daemon off;"

Nginx 镜像运行的命令

-g "daemon off;"是关闭 nginx 后台运行；nginx 默认是以后台模式启动的，Docker 未执行自定义的 CMD 之前，nginx 的 pid 是 1，执行到 CMD 之后，nginx 就在后台运行，bash 或 sh 脚本的 pid 变成了 1。所以一旦执行完自定义 CMD，nginx 容器也就退出了。为了保持 nginx 的容器不退出，应该关闭 nginx 后台运行

```
^C[root@localhost dockerfile]# docker run -d -p 80 cnetos-nginx nginx -g "daemon off;"
58c9e2d576dcda2cefb42ba7b33195402b16239e65d176c95a31607c9a016385
```

6.查看容器运行

docker ps -l

镜像运行后映射的端口是 49154

```
[root@centos7test ~]# docker ps -l
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                   NAMES
0405dc7e8b0a   tomcat9   "nginx -g 'daemon of…"   18 seconds ago   Up 17 seconds   0.0.0.0:49154->80/tcp, :::49154->80/tcp   frosty_leakey
[root@centos7test ~]#
```

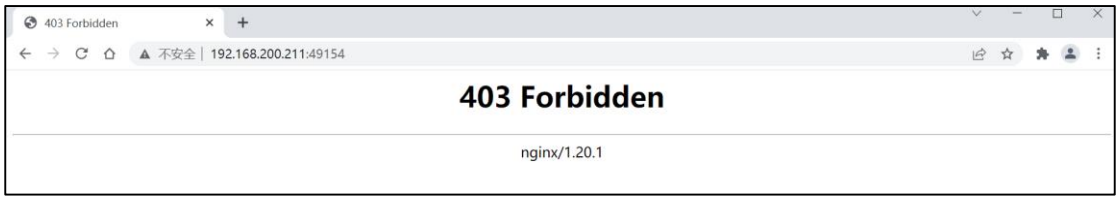
7.验证

宿主机里运行

curl 127.0.0.1:49154

```
[root@centos7test ~]# curl 127.0.0.1:49154
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.20.1</center>
</body>
</html>
[root@centos7test ~]#
```

192.216.5.211 是宿主机的 IP 地址



下图是基础镜像正常结果，上图不正常进入容器到 nginx 的发布目录/usr/share/nginx/html 会发现 index.html 异常，删除 index.html 文件，再新建 index.html 就正常了。

8. 查看镜像 cnetos-nginx 运行的历史记录

docker history cnetos-nginx

```
[root@localhost dockerfile]# docker history cnetos-nginx
IMAGE          CREATED         CREATED BY                                      SIZE      COMMENT
51ae5c37de3f   About an hour ago   /bin/sh -c #(nop)  EXPOSE 80                 0B
acf981475521   About an hour ago   /bin/sh -c yum install -y nginx              74.3MB
027b13b1f7bf   About an hour ago   /bin/sh -c #(nop)  MAINTAINER CHANGAN "123@Q鈥  0B
831691599b88   6 weeks ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]         0B
<missing>      6 weeks ago         /bin/sh -c #(nop)  LABEL org.label-schema.sc鈥  0B
<missing>      6 weeks ago         /bin/sh -c #(nop) ADD file:84700c11fcc969ac0鈥  215MB
[root@localhost dockerfile]#
```

（六）Dockerfile 生成新的镜像 3（ubuntu1）

1.建立 Dockerfile 文件

vi Dockerfile

```
[root@centos7test test]# vi Dockerfile

FROM ubuntu
CMD echo "this is a test"
```

2.生成镜像

docker build --tag hello .

```
[root@centos7test test]# docker build --tag hello .
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM ubuntu
 ---> 54c9d81cbb44
Step 2/2 : CMD echo "this is a test"
 ---> Running in ea7270a756c9
Removing intermediate container ea7270a756c9
 ---> cf08a29e4978
Successfully built cf08a29e4978
Successfully tagged hello:latest
```

3.运行镜像

docker run hello

```
[root@centos7test test]# docker run hello
this is a test
[root@centos7test test]#
```

（六）Dockerfile 生成新的镜像 3（ubuntu2）

1.制作可执行脚本文件

（1）# vi hello

```
[root@centos7test test]# vi hello

#!/bin/sh
echo 'hello word'
```

（2）赋予执行权限

chmod a+x hello

2. 建立 Dockerfile 文件

vi Dockerfile

```
[root@centos7test test]# vi Dockerfile

FROM ubuntu
ADD hello /
CMD ["/hello"]
```

3.生成镜像文件

docker build --tag hellotest .

```
[root@centos7test test]# docker build --tag hellotest .
Sending build context to Docker daemon  3.072kB
Step 1/3 : FROM ubuntu
 ---> 54c9d81cbb44
Step 2/3 : ADD hello /
 ---> f80f6ed2051f
Step 3/3 : CMD ["/hello"]
 ---> Running in 22fa9c4e1691
Removing intermediate container 22fa9c4e1691
 ---> 0ba703df95fa
Successfully built 0ba703df95fa
Successfully tagged hellotest:latest
```

4.运行镜像

docker run hellotest

```
[root@centos7test test]# docker run hellotest
hello word
```

（六）Dockerfile 生成新的镜像 2（ubuntu-nginx）

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>              07f15817012f        4 minutes ago       126MB
centos              latest              470671670cac        3 days ago          237MB
ubuntu              latest              ccc6e87d482b        5 days ago          64.2MB
nginx               latest              c7460dfcab50        11 days ago         126MB
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        3 weeks ago         1.22MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
[root@localhost ~]#
```

mkdir www

cd www

touch Dockerfile 必须是 Dockerfile，而且 D 大写

ls

```
[root@localhost www]# ls
Dockerfile
```

vi Dockerfile

FROM ubuntu

MAINTAINER WWW

RUN apt-get update

RUN apt-get install -y nginx

RUN echo "this is home" > /var/www/html/index.html

EXPOSE 80

```
FROM ubuntu
MAINTAINER WWW
RUN apt-get update
RUN apt-get install -y nginx
RUN echo "thi is home" > /var/www/html/index.html
EXPOSE 80
```

docker build -t="ceshi/static_web" . //注意后面有个点，表示在当前目录，-t 表示构建一个名称

```
[root@localhost static_web]# docker build -t="ceshi/static_web" .
Sending build context to Docker daemon  2.048kB
Step 1/6 : FROM ubuntu
 ---> ccc6e87d482b
Step 2/6 : MAINTAINER WWW
 ---> Running in 2238ff6f20dc
Removing intermediate container 2238ff6f20dc
 ---> 0a82d9199b58
Step 3/6 : RUN apt-get update
 ---> Running in 05d99a1928d7
```

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ceshi/static_web    latest              3f89136d2a7e        26 minutes ago      152MB
<none>              <none>              cb235a472003        36 minutes ago      92MB
<none>              <none>              31ea0e7eaade        44 minutes ago      152MB
<none>              <none>              07f15817012f        55 minutes ago      126MB
centos              latest              470671670cac        3 days ago          237MB
ubuntu              latest              ccc6e87d482b        5 days ago          64.2MB
nginx               latest              c7460dfcab50        11 days ago         126MB
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        3 weeks ago         1.22MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
[root@localhost ~]#
```

docker run -d -it ceshi/static_web

```
[root@centos7test nginx]# docker run -d -it ceshi/static_web
8ad7bcda935a7d6fac8f09c836324a4450e73737da8ae5492049c5d9bcc8ebed
[root@centos7test nginx]#
```

docker ps -l

```
[root@centos7test nginx]#  docker ps -l
CONTAINER ID   IMAGE              COMMAND   CREATED              STATUS              PORTS     NAMES
8ad7bcda935a   ceshi/static_web   "bash"    About a minute ago   Up About a minute   80/tcp    epic_bose
[root@centos7test nginx]#
```

（六）Dockerfile 生成新的镜像 3（ubuntu-vnc）

1.建立一个目录用作构建上下文，并切换到该目录。

2.在该目录中创建 Dockerfile 文件。

```
FROM ubuntu
# 安装用于创建图形化界面的 VNC 和 xvfb，以及浏览器 Firefox
RUN apt-get update && apt-get install -y x11vnc xvfb firefox
RUN mkdir ~/.vnc
# 设置 VNC 登录密码
RUN x11vnc -storepasswd 1234 ~/.vnc/passwd
# 自动启动 Firefox
RUN bash -c 'echo "firefox" >> /.bashrc'
EXPOSE 5900
CMD    ["x11vnc", "-forever", "-usepw", "-create"]
```

3.使用 docker build 命令构建镜像。

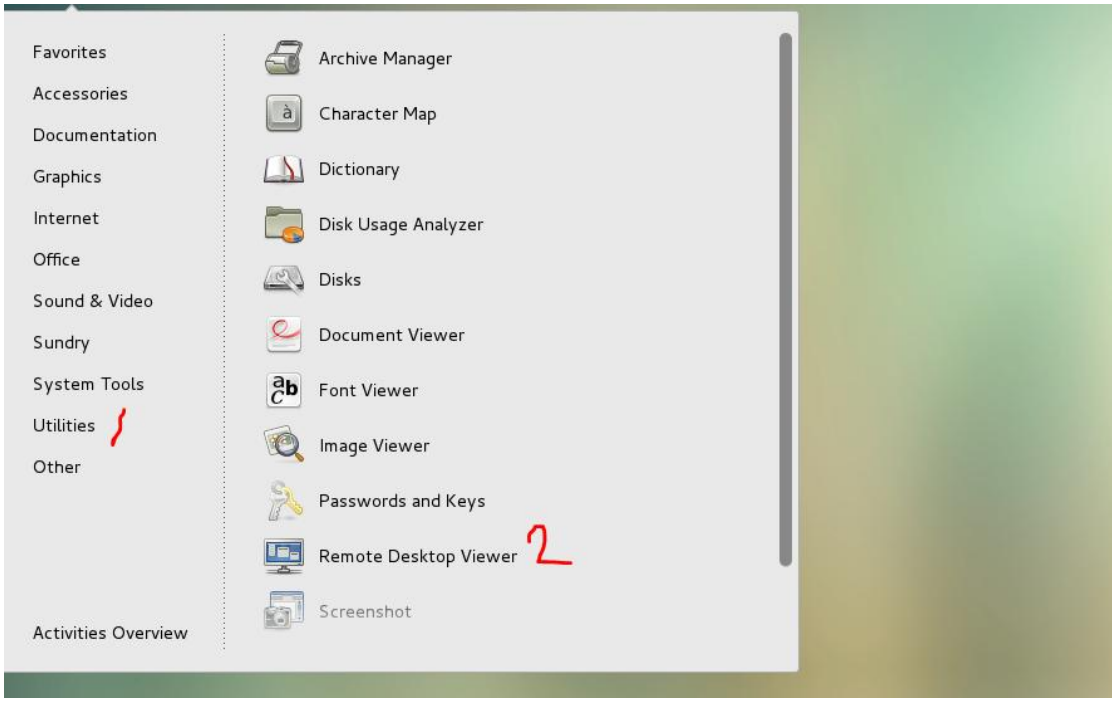
```
[root@host1 fx-vnc]# docker build --tag fx-vnc .
Sending build context to Docker daemon    2.048kB
……
```

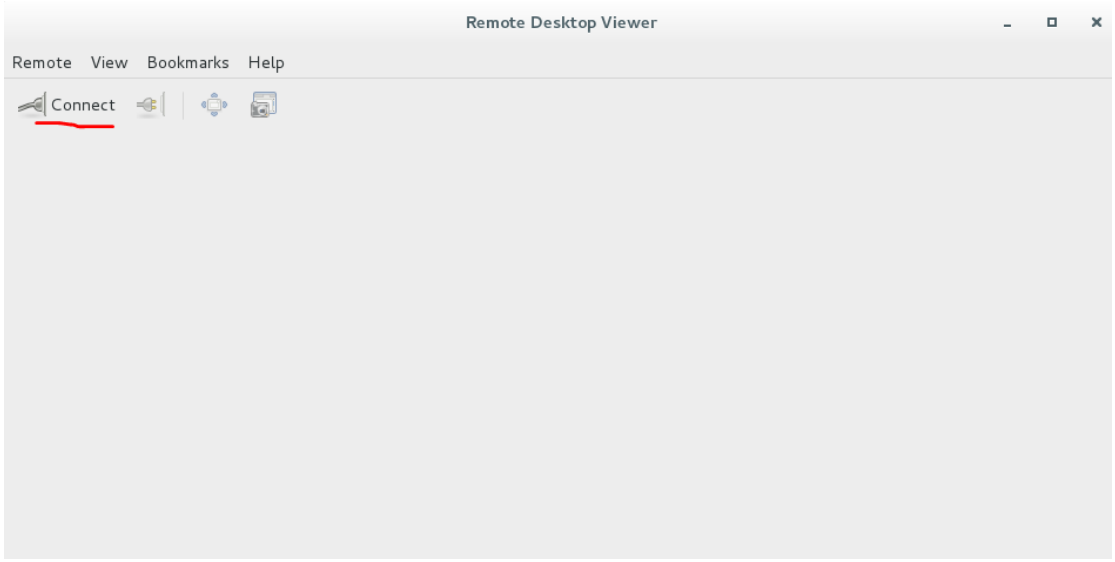
4.运行这个新镜像启动容器。

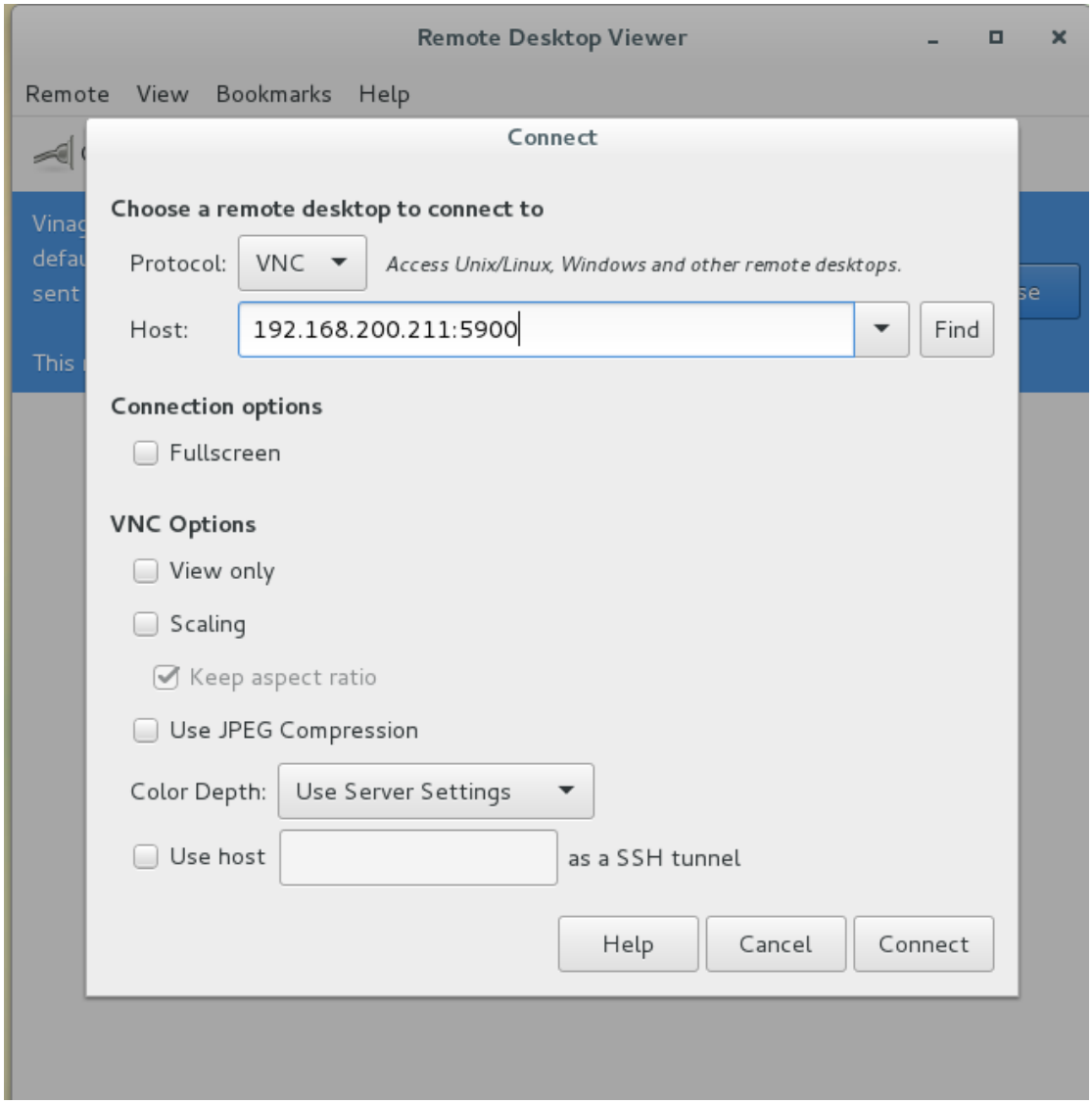
```
[root@host1 fx-vnc]# docker run --rm -p 5900:5900 fx-vnc
```

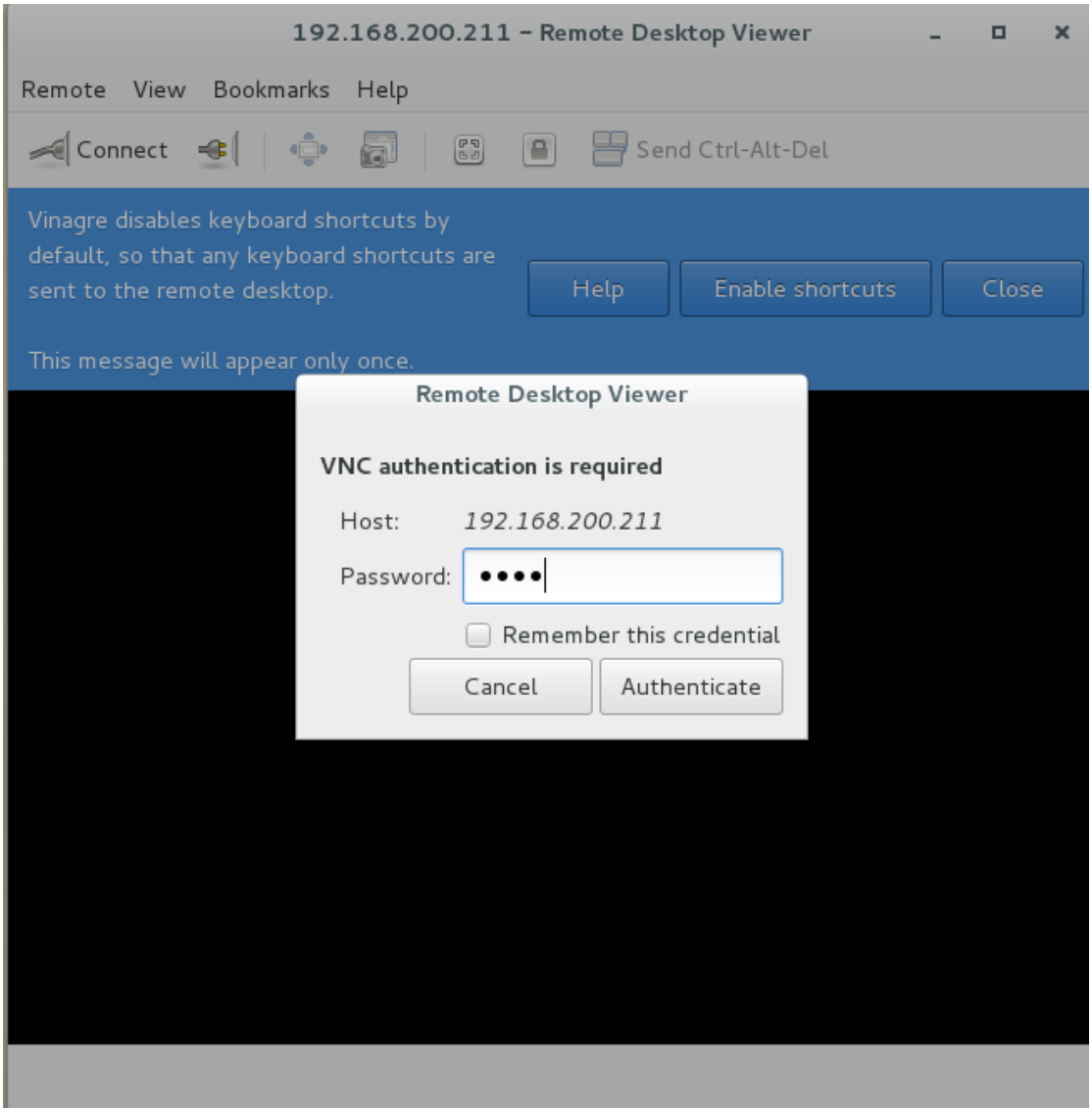
```
[root@centos7test nginx]# docker run --rm -p 5900:5900 fx-vnc
25/02/2022 12:54:46 -usepw: found /root/.vnc/passwd
25/02/2022 12:54:46 x11vnc version: 0.9.16 lastmod: 2019-01-05  pid: 1
25/02/2022 12:54:46
25/02/2022 12:54:46 wait_for_client: WAIT:cmd=FINDCREATEDISPLAY-Xvfb
25/02/2022 12:54:46
25/02/2022 12:54:46 initialize_screen: fb_depth/fb_bpp/fb_Bpl 24/32/2560
25/02/2022 12:54:46
25/02/2022 12:54:46 Autoprobing TCP port
25/02/2022 12:54:46 Autoprobing selected TCP port 5900
25/02/2022 12:54:46 Autoprobing TCP6 port
25/02/2022 12:54:46 Autoprobing selected TCP6 port 5900
25/02/2022 12:54:46 listen6: bind: Address already in use
25/02/2022 12:54:46 Not listening on IPv6 interface.
25/02/2022 12:54:46

The VNC desktop is:      8a8097958e93:0
PORT=5900
```

5.设置要连接的远程桌面

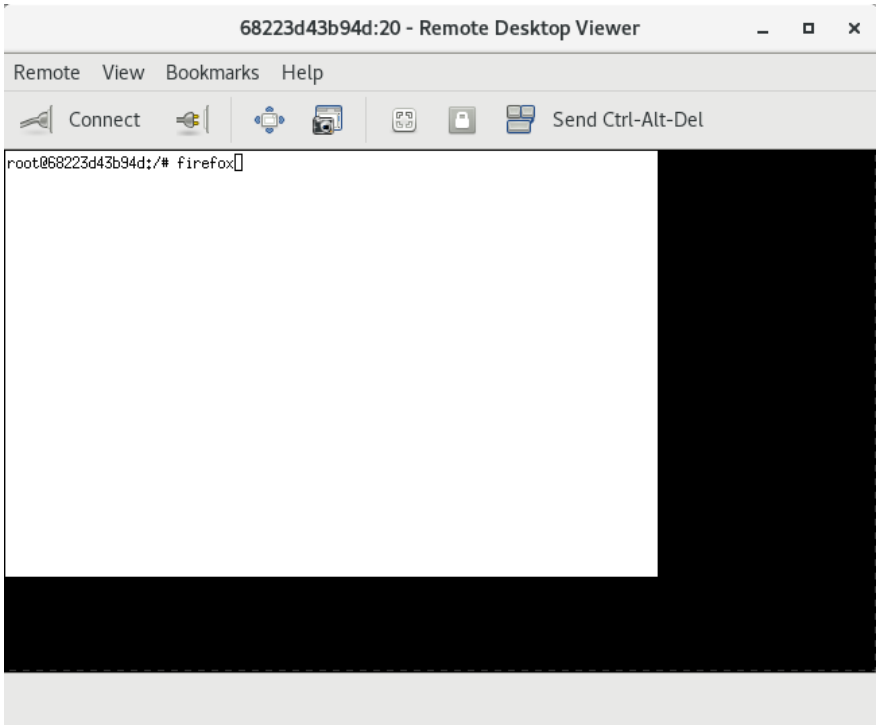

Favorites
Accessories
Documentation
Graphics
Internet
Office
Sound & Video
Sundry
System Tools
Utilities
Other
Activities Overview
Archive Manager
Character Map
Dictionary
Disk Usage Analyzer
Disks
Document Viewer
Font Viewer
Image Viewer
Passwords and Keys
Remote Desktop Viewer
Screenshot
1
2


Remote Desktop Viewer
Remote
View
Bookmarks
Help
Connect

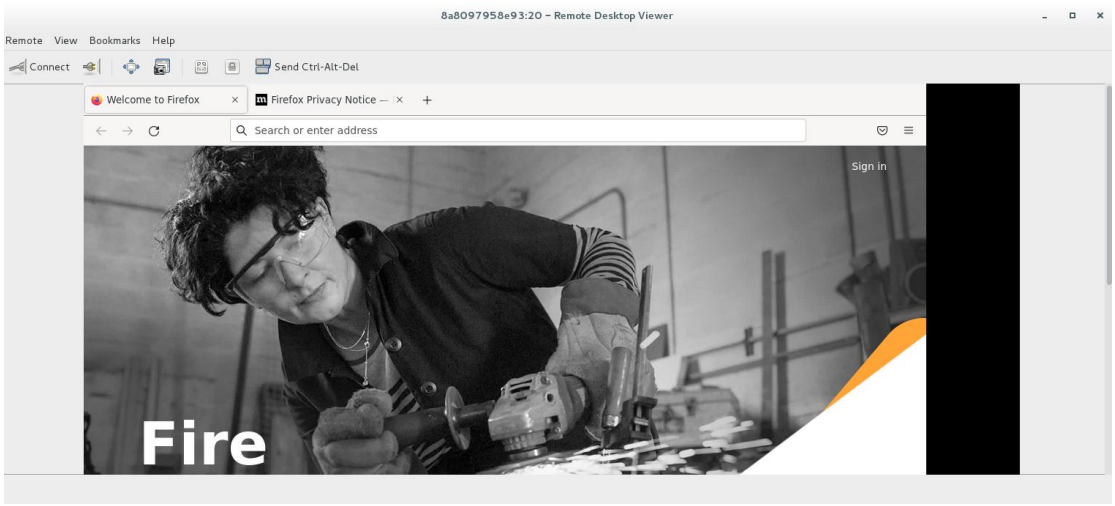

Remote Desktop Viewer
Remote View Bookmarks Help
Connect
Choose a remote desktop to connect to
Protocol: VNC
Access Unix/Linux, Windows and other remote desktops.
Host: 192.168.200.211:5900
Find
Connection options
Fullscreen
VNC Options
View only
Scaling
Keep aspect ratio
Use JPEG Compression
Color Depth: Use Server Settings
Use host
as a SSH tunnel
Help
Cancel
Connect


192.168.200.211 – Remote Desktop Viewer
Remote View Bookmarks Help
Connect
Send Ctrl-Alt-Del
Vinagre disables keyboard shortcuts by default, so that any keyboard shortcuts are sent to the remote desktop.
Help
Enable shortcuts
Close
This message will appear only once.
Remote Desktop Viewer
VNC authentication is required
Host: 192.168.200.211
Password:
Remember this credential
Cancel
Authenticate

登录远程桌面



远程桌面上出现浏览器界面



（六）Dockerfile 生成新的镜像 3（node-bulletin-board）

1.安装 git

yum install -y epel-release

yum install -y git

```
[root@centos7test ~]# yum install -y git
Loaded plugins: fastestmirror, langpacks
epel/x86_64/metalink                                                                                    | 9.7 kB  00:00:00
https://mirrors.tuna.tsinghua.edu.cn/epel/7/x86_64/repodata/repomd.xml: [Errno 14] curl#60 - "The certificate issuer's certificate has expired.  Ch
eck your system date and time."
Trying other mirror.
It was impossible to connect to the CentOS servers.
This could mean a connectivity issue in your environment, such as the requirement to configure a proxy,
or a transparent proxy that tampers with TLS security, or an incorrect system clock.
Please collect information about the specific failure that occurs in your environment,
using the instructions in: https://access.redhat.com/solutions/1527033 and create a bug on https://bugs.centos.org/

https://mirrors.ustc.edu.cn/epel/7/x86_64/repodata/repomd.xml: [Errno 14] curl#60 - "Peer's Certificate issuer is not recognized."
Trying other mirror.
epel                                                                                                    | 4.7 kB  00:00:00
(1/3): epel/x86_64/group_gz                                                                             |  96 kB  00:00:00
(2/3): epel/x86_64/primary_db                                                                           | 7.0 MB  00:00:07
epel/x86_64/updateinfo       FAILED
https://mirror.lzu.edu.cn/epel/7/x86_64/repodata/23690ea9ff214d6a0efb611681e2ca9120a49c892979ed3be0910f3d9a16e419-updateinfo.xml.bz2: [Errno 14] cu
rl#7 - "Failed to connect to 2001:da8:c000::160: Network is unreachable"
Trying other mirror.
(3/3): epel/x86_64/updateinfo                                                                           | 1.1 MB  00:00:00
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * epel: mirrors.bfsu.edu.cn
```

2. 下载程序

git clone -b v1 https://github.com/docker-training/node-bulletin-board.git

```
[root@centos7test opt]# git clone -b v1 https://github.com/docker-training/node-bulletin-board.git
Cloning into 'node-bulletin-board'...
remote: Enumerating objects: 213, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 213 (delta 8), reused 17 (delta 4), pack-reused 190
Receiving objects: 100% (213/213), 197.64 KiB | 0 bytes/s, done.
Resolving deltas: 100% (90/90), done.
[root@centos7test opt]# ls
containerd  docker-file  node-bulletin-board  rh
[root@centos7test opt]#
```

3.进入下载程序目录

cd node-bulletin-board/

cd bulletin-board-app/

```
[root@centos7test ~]# cd node-bulletin-board/
[root@centos7test node-bulletin-board]# cd bulletin-board-app/
[root@centos7test bulletin-board-app]#
```

4.查看程序已经编辑好的 Dockerfile 文件

vi Dockerfile

```
[root@centos7test bulletin-board-app]# vi Dockerfile

FROM node:6.11.5

WORKDIR /usr/src/app
COPY package.json .
RUN npm install
COPY . .

CMD [ "npm", "start" ]

~
~
```

5.生成镜像

docker build -t bulletinboard:v1 .

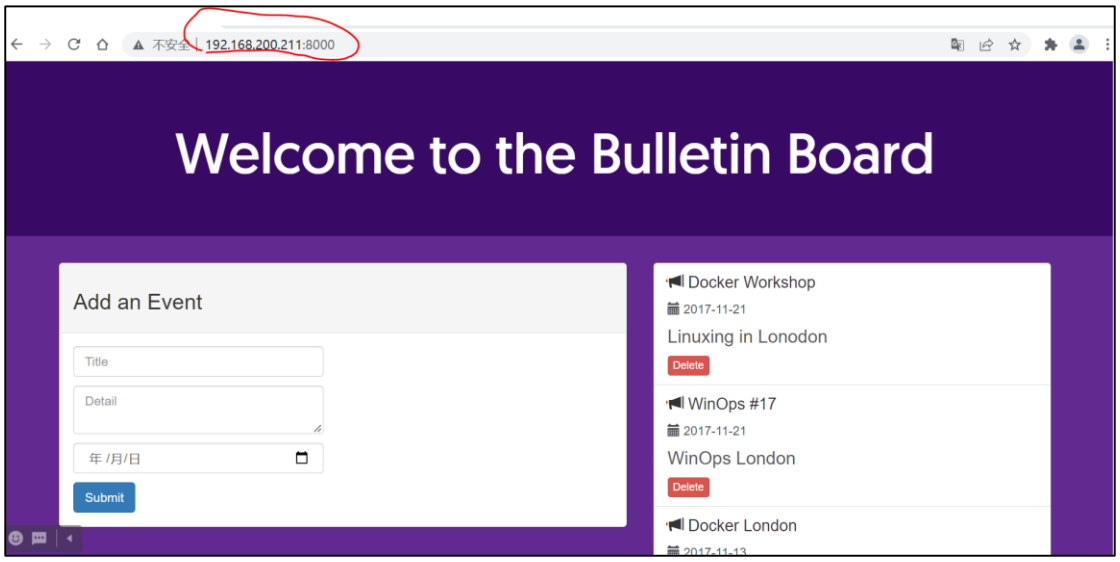
```
[root@centos7test bulletin-board-app]# docker build -t bulletinboard:v1 .
Sending build context to Docker daemon  45.57kB
Step 1/6 : FROM node:6.11.5
 ---> 852391892b9f
Step 2/6 : WORKDIR /usr/src/app
 ---> Running in 47e185fdfc7e
Removing intermediate container 47e185fdfc7e
 ---> a79d7af30ea6
Step 3/6 : COPY package.json .
 ---> e4d2d17ec0c4
Step 4/6 : RUN npm install
 ---> Running in ff1332727e7a

> ejs@2.7.4 postinstall /usr/src/app/node_modules/ejs
> node ./postinstall.js
```

6.运行镜像

docker run -p 8000:8080 -d bulletinboard:v1

```
[root@centos7test ~]# docker run -p 8000:8080 -d bulletinboard:v1
4b236add993dabcd33a365c50aa960f7a215e6264fbe6f8f5a928da8f6aaf6df
[root@centos7test ~]#
```

7.客户端访问



（七）删除镜像

1．若是带有不是 latest 标记一定要加上标记删除

docker rmi 镜像名:[标记]

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
new_centos1         latest              8d2df9544ce0        7 hours ago         297MB
<none>              <none>              44593bea2015        7 hours ago         165MB
centos              latest              470671670cac        2 days ago          237MB
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
[root@localhost ~]#
```

docker rmi new_centos

```
[root@localhost ~]# docker rmi new_centos1
Untagged: new_centos1:latest
Deleted: sha256:8d2df9544ce0ba7cb8d57f7cc7a6a66ac81024f04435fa33fa2df2d760b9a00f
Deleted: sha256:3844e939ace85d39b1829a0788af943caa1b19ca88d64b4a2ff2f63581ee0852
[root@localhost ~]#
```

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>              44593bea2015        7 hours ago         165MB
centos              latest              470671670cac        2 days ago          237MB
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
[root@localhost ~]#
```

2.

```
[root@centos7test ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
root_web            latest              dab3c32eaaf9        12 hours ago        964MB
postgres            latest              d7337c283715        5 days ago          376MB
python              3                   178dcaa62b39        5 days ago          917MB
postgres            9.6.23              8f39a059063c        4 months ago        200MB
hello-world         latest              feb5d9fea6a5        5 months ago        13.3kB
[root@centos7test ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@centos7test ~]# docker rmi dab3c32eaaf9
Error response from daemon: conflict: unable to delete dab3c32eaaf9 (must be forced) - image is being used by stopped container 26d56eec3f06
[root@centos7test ~]#
```

```
[root@centos7test ~]# docker rm 26d56eec3f06
26d56eec3f06
[root@centos7test ~]# docker rmi dab3c32eaaf9
Error response from daemon: conflict: unable to delete dab3c32eaaf9 (must be forced) - image is being used by stopped container e86a1726519b
[root@centos7test ~]# docker rm e86a1726519b
e86a1726519b
[root@centos7test ~]# docker rmi dab3c32eaaf9
Untagged: root_web:latest
Deleted: sha256:dab3c32eaaf9798b6dc253439040d363605713547c68c2a4b998a196e0c980d6
Deleted: sha256:e8f4a2871333ff0a742bf5e439706cb44d5d5a75e55d9794c5026fd845f3db99
Deleted: sha256:43fe5d1bbc5156b81e76cc2b3fa4be0a6961cd2475aa659a3955b47d1fb407f0
Deleted: sha256:881dd2bee3f2bb8ba44caa3162aea31d59b1244a6962369fce39fa4326e125d6
Deleted: sha256:dad22a680fcc5de973120f752a15afdbf4bbf30a8db0d38f8e73bfe026d2baf6
Deleted: sha256:99a69bdfb49aa1fb27f3f824b8516853 6b521d22688909d128f3d3bf46a35731
Deleted: sha256:8831163de83f87c52c0d748192408 1f7472e22d94cf74c276a99c442ebddea45
Deleted: sha256:df1ed35541ff3d4fec399219b66fff9eede85c020084a1bfd6e3c3b3f8f1f5d8
Deleted: sha256:aec3e53ec4378108fbd13bb424203967 9613a794a4e7834882a8d9a749538d67
Deleted: sha256:55b25e85db492d73e9617cae12d0f0edb23cd4cc6bb9103b9580ef7fda4555a8
[root@centos7test ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
postgres            latest              d7337c283715        5 days ago          376MB
python              3                   178dcaa62b39        5 days ago          917MB
postgres            9.6.23              8f39a959063c        4 months ago        200MB
hello-world         latest              feb5d9fea6a5        5 months ago        13.3kB
[root@centos7test ~]#
```

（八）导出镜像

目的把镜像从一台主机复制到另一台

Docker 默认的存储目录位于/var/lib/docker

Docker save -o　导出文件名 本地镜像

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              470671670cac        9 days ago          237MB
nginx               latest              c7460dfcab50        2 weeks ago         126MB
httpd               latest              c2aa7e16edd8        4 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        4 weeks ago         1.22MB
registry            latest              f32a97de94e1        10 months ago       25.8MB
hello-world         latest              fce289e99eb9        13 months ago       1.84kB
[root@localhost ~]# docker save -o daochu hello-world
[root@localhost ~]#
```

docker save -o daochu hello-world

```
[root@localhost ~]# docker save -o daochu hello-world
[root@localhost ~]#
```

查看结果

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# ls
anaconda-ks.cfg  ceshi  daochu  sample
[root@localhost ~]#
```

（九）导入镜像

docker load <导出文件名

或者

docker load -I 导出文件名

docker load -i daochu

```
[root@localhost ~]# docker load -i daochu
Loaded image: hello-world:latest
[root@localhost ~]#
```

（九）导入镜像

ls /var/lib/docker

这个目录是 docker 工作目录

```
[root@localhost ~]# ls /var/lib/docker
builder   containers  network   plugins   swarm   trust
buildkit  image       overlay2  runtimes  tmp     volumes
[root@localhost ~]#
```

Docker images 使用

docker images –help

```
[root@centos71611 ~]# docker images --help

Usage:  docker images [OPTIONS] [REPOSITORY[:TAG]]

List images

Options:
  -a, --all             Show all images (default hides intermediate images)
      --digests         Show digests
  -f, --filter filter   Filter output based on conditions provided
      --format string   Pretty-print images using a Go template
      --no-trunc        Don't truncate output
  -q, --quiet           Only show numeric IDs
[root@centos71611 ~]#
```

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
busybox             latest              018c9d7b792b        5 days ago
1.22MB
httpd               latest              9d2a0c6e5b57        10 days ago
166MB
nginx               latest              8cf1bfb43ff5        10 days ago
132MB
centos              latest              831691599b88        6 weeks ago
215MB
hello-world         latest              bf756fb1ae65        7 months ago
13.3kB
```

docker images -a

显示所有镜像

```
[root@localhost ~]# docker images -a
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
busybox             latest              018c9d7b792b        5 days ago          1.22MB
httpd               latest              9d2a0c6e5b57        10 days ago         166MB
nginx               latest              8cf1bfb43ff5        10 days ago         132MB
centos              latest              831691599b88        6 weeks ago         215MB
hello-world         latest              bf756fb1ae65        7 months ago        13.3kB
[root@localhost ~]#
```

docker images -q

只显示镜像的 id 号

```
[root@localhost ~]# docker images -q
018c9d7b792b
9d2a0c6e5b57
8cf1bfb43ff5
831691599b88
bf756fb1ae65
[root@localhost ~]#
```

docker images --no-trunc

```
[root@localhost ~]# docker images --no-trunc
REPOSITORY          TAG                 IMAGE ID                                                                  CREATE
D                   SIZE
busybox             latest              sha256:018c9d7b792b4be80095d957533667279843acf9a46c973067c8d1dff31ea8b4   5 days
 ago                1.22MB
httpd               latest              sha256:9d2a0c6e5b5714303c7b72793311d155b1652d270a785c25b88197069ba78734   10 day
s ago               166MB
nginx               latest              sha256:8cf1bfb43ff5d9b05af9b6b63983440f137c6a08320fa7592197c1474ef30241   10 day
s ago               132MB
centos              latest              sha256:831691599b88ad6cc2a4abbd0e89661a121aff14cfa289ad840fd3946f274f1f   6 week
s ago               215MB
hello-world         latest              sha256:bf756fb1ae65adf866bd8c456593cd24beb6a0a061dedf42b26a993176745f6b   7 mont
hs ago              13.3kB
[root@localhost ~]#
```

docker images centos

```
[root@localhost ~]# docker images centos
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              831691599b88        6 weeks ago         215MB
[root@localhost ~]#
```

四、Docker 容器

容器是镜像的运行实例。不同的是，镜像是静态的只读文件，而容器带有运行时需要的可写文件层；同时，容器中的应用进程处于运行状态。

（一）停止运行容器

docker stop d5a4eb50a064

容器的 id 号（CONTAINER ID）

或者

docker stop dazzling_gauss

容器的名称（NAMES）

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
d5a4eb50a064        httpd               "httpd-foreground"  2 hours ago         Up 2 hours          0.0.0.0:80->80/tcp  dazzling_gauss
[root@localhost ~]# docker stop d5a4eb50a064
d5a4eb50a064
```

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@localhost ~]# docker stop d5a4eb50a064
```

（二）让容器一直运行

docker run centos /bin/bash -c "while true;do sleep 1;done "

```
[root@localhost ~]# docker run centos /bin/bash -c "while true;do sleep 1;done "
```

打开终端另一个窗口，查看

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
dbe81c50658b        centos              "/bin/bash -c 'while…"   19 seconds ago      Up 18 seconds                           determined_visve
svaraya
[root@localhost ~]#
```

这里没有加参数，若是加上-itd 或者-d 容器就能一直运行

（三）进入容器

镜像运行后，再进入容器

方法 1

docker run -it centos /bin/bash

```
[root@localhost ~]# docker run -it centos /bin/bash
[root@db968d3a1319 /]# ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
7: eth0@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
;@db968d3a1319:/[root@db968d3a1319 /]#
```

按下 ctrl+q 然后 ctrl+p 退出

```
;@db968d3a1319:/[root@db968d3a1319 /]# [root@localhost ~]#
[root@localhost ~]#
```

docker ps -l 查看容器还在运行

```
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
 PORTS               NAMES
db968d3a1319        centos              "/bin/bash"         About a minute ago   Up About a minute
                    modest_wright
[root@localhost ~]# _
```

docker attach db968d3a1319 再次进入容器

```
[root@localhost ~]# docker attach db968d3a1319
[root@db968d3a1319 /]# ls
bin  etc   lib    lost+found  mnt  proc  run   srv  tmp  var
dev  home  lib64  media       opt  root  sbin  sys  usr
;@db968d3a1319:/[root@db968d3a1319 /]# _
```

exit 方式退出容器

```
;@db968d3a1319:/[root@db968d3a1319 /]# exit
exit
[root@localhost ~]# _
```

docker ps -l 查看容器不运行

```
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
        PORTS               NAMES
db968d3a1319        centos              "/bin/bash"         2 minutes ago       Exited (0) 24 second
s ago                       modest_wright
[root@localhost ~]# _
```

方法 2

```
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
db968d3a1319        centos              "/bin/bash"         54 minutes ago      Up 29 seconds
                    modest_wright
```

```
[root@localhost ~]# docker exec -it db968d3a1319 /bin/bash
[root@db968d3a1319 /]# _
```

attach 和 exec 方式区别

- attach 直接进入容器启动命令的终端，不会启动新的进程。
- exec 则是在容器中打开**新的终端**，并且可以启动新的进程。
- 如果想直接在终端中查看启动命令的输出，用 attach；其他情况使用 exec
- exec 方式进入 exit 退出不会导致容器终止运行
- attach 方式进入所有的**终端输出是同样的内容**

大多数情况，可以直接使用 docker exec 命令。

（四）暂停与恢复容器运行

1.暂停容器运行

docker pause 41b2ef9727f4

```
[root@localhost ~]# docker pause 41b2ef9727f4
41b2ef9727f4
[root@localhost ~]#
```

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
41b2ef9727f4        centos              "/bin/bash -c 'while…"   About a minute ago  Up About a minute (Paused)                      pensi
e_shamir
[root@localhost ~]#
```

2. 恢复容器运行

docker unpause 41b2ef9727f4

```
[root@localhost ~]# docker unpause 41b2ef9727f4
41b2ef9727f4
```

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
41b2ef9727f4        centos              "/bin/bash -c 'while…"   3 minutes ago       Up 3 minutes                            pensive_shamir
[root@localhost ~]#
```

（五）杀死容器进程

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
f973c3a31ee1        httpd               "httpd-foreground"       4 hours ago         Up 4 hours          80/tcp              cocky_haibt
[root@localhost ~]#
```

docker kill f973c3a31ee1

```
[root@localhost ~]# docker kill f973c3a31ee1
f973c3a31ee1
[root@localhost ~]#
```

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@localhost ~]#
```

（五）删除容器（曾经运行过）

1.查看曾经运行过的容器

docker ps -a

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
41b2ef9727f4        centos              "/bin/bash -c 'while…"   21 minutes ago      Up 21 minutes                                   pensi
ve_shamir
84844ea37d67        centos              "/bin/bash -c 'while…"   25 minutes ago      Exited (137) 21 minutes ago                     elast
ic_villani
744afcbaa0a6        centos              "/bin/bash"              25 minutes ago      Exited (0) 25 minutes ago                       nervo
us_boyd
96f706d08979        centos              "/bin/sh"                25 minutes ago      Exited (0) 25 minutes ago                       festi
ve_bouman
51d2563fd452        centos              "/bin/bash -c 'while…"   39 minutes ago      Exited (137) 27 minutes ago                     eloqu
ent_goldberg
dbe81c50658b        centos              "/bin/bash -c 'while…"   53 minutes ago      Exited (137) 49 minutes ago                     deter
mined_visvesvaraya
0c405917d196        centos              "/bin/bash"              57 minutes ago      Exited (0) 57 minutes ago                       angry
_allen
fa7b76f484d1        centos              "pwd"                    57 minutes ago      Exited (0) 57 minutes ago                       crazy
_chatelet
2f177d1c704f        centos              "/bin/bash"              58 minutes ago      Exited (0) 58 minutes ago                       thirs
ty_benz
2767d69badb3        centos              "/bin/bash"              58 minutes ago      Exited (0) 58 minutes ago                       cocky
_perlman
a44466a44cdb        centos              "/bin/bash"              2 hours ago         Exited (0) About an hour ago                    crank
y_ride
1a67d3e615ce        centos              "/bin/bash"              2 hours ago         Exited (127) 2 hours ago                        suspi
cious_tesla
85bbe66b4d21        hello-world         "/hello"                 2 hours ago         Exited (0) 2 hours ago                          crank
y_chaum
d5a4eb50a064        httpd               "httpd-foreground"       3 hours ago         Exited (0) About an hour ago                    dazzl
ing_gauss
[root@localhost ~]#
```

2.删除容器

docker rm 84844ea37d67

```
[root@localhost ~]# docker rm 84844ea37d67
84844ea37d67
[root@localhost ~]#
```

3.批量删除曾经运行过的容器

docker rm -v $(docker ps -aq -f status=exited)

```
[root@localhost ~]# docker rm -v $(docker ps -aq -f status=exited)
744afcbaa0a6
96f706d08979
51d2563fd452
dbe81c50658b
0c405917d196
fa7b76f484d1
2f177d1c704f
2767d69badb3
a44466a44cdb
1a67d3e615ce
85bbe66b4d21
d5a4eb50a064
[root@localhost ~]#
```

（五）删除容器（正在运行）

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                  NAMES
05f56e4d8d00        nginx               "nginx -g 'daemon of…"   24 minutes ago      Up 24 minutes       0.0.0.0:80->80/tcp     beautiful_agnesi
[root@localhost ~]#
```

docker rm -f 05f56e4d8d00

-f 是强制删除

```
[root@localhost ~]# docker rm -f 05f56e4d8d00
05f56e4d8d00
[root@localhost ~]#
```

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@localhost ~]#
```

（六）创建容器，再执行

1.查看镜像

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
new_centos1         latest              8d2df9544ce0        3 hours ago         297MB
new_centos          latest              89a48cacac53        3 hours ago         237MB
<none>              <none>              44593bea2015        3 hours ago         165MB
centos              latest              470671670cac        2 days ago          237MB
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
```

2.建立容器

docker create httpd

```
[root@localhost ~]# docker create httpd
f973c3a31ee1bb3a72ce7ca519cfb04e021e46c2fb64c54d320d12bca2d31658
```

3.查看容器线程

docker ps -a

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
f973c3a31ee1        httpd               "httpd-foreground"       16 seconds ago      Created                                         cocky_haibt
41b2ef9727f4        centos              "/bin/bash -c 'while…"   2 hours ago         Exited (137) About a minute ago                 pensive_shamir
```

4.开始运行容器

docker start f973c3a31ee1

```
[root@localhost ~]# docker start f973c3a31ee1
f973c3a31ee1
```

3.查看容器线程

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
f973c3a31ee1        httpd               "httpd-foreground"  About a minute ago  Up 7 seconds        80/tcp              cocky_haibt
[root@localhost ~]#
```

（六）容器导出

导出容器是指将一个已经创建（不管此时容器是否处于运行状态）的容器到一个文件。

docker export　容器 id>　导出文件名

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
4b1a7828f5f5        busybox             "sh"                     7 seconds ago       Up 6 seconds                                 recursing_kirch
1126d76f16bd        registry            "/entrypoint.sh /etc…"   3 days ago          Up 49 minutes       0.0.0.0:5000->5000/tcp   registry
[root@localhost ~]#
```

[root@localhost ~]# docker export 4b1a7828f5f5>rongqidaochu

```
[root@localhost ~]# docker export 4b1a7828f5f5>rongqidaochu
```

结果

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# ls
anaconda-ks.cfg  ceshi  daochu  rongqidaochu  sample
[root@localhost ~]#
```

（七）容器导入为镜像

可以将导出的容器导入变成镜像。

cat　导出文件名|docker import - 生成的镜像名:标记

cat rongqidaochu | docker import - xinjingxiang:new

```
[root@localhost ~]# cat rongqidaochu | docker import - xinjingxiang:new
sha256:7e8126329110793d2852f4675ee93b218ba75459315a5368d36fe6ff63633d57
[root@localhost ~]#
```

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
xinjingxiang        new                 7e8126329110        54 seconds ago      1.22MB
centos              latest              470671670cac        9 days ago          237MB
nginx               latest              c7460dfcab50        2 weeks ago         126MB
httpd               latest              c2aa7e16edd8        4 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        4 weeks ago         1.22MB
registry            latest              f32a97de94e1        10 months ago       25.8MB
hello-world         latest              fce289e99eb9        13 months ago       1.84kB
```

（七）查看容器输出信息

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
05f56e4d8d00        nginx               "nginx -g 'daemon of…"   24 minutes ago      Up 24 minutes       0.0.0.0:80->80/tcp  beautiful_agnesi
[root@localhost ~]#
```

docker logs 05f56e4d8d00

```
[root@localhost ~]# docker logs 05f56e4d8d00
192.168.200.1 - - [21/Jan/2020:10:26:10 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Firefox/72.0" "
"
2020/01/21 10:26:10 [error] 6#6: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 192.168.200.1, server: localho
t, request: "GET /favicon.ico HTTP/1.1", host: "192.168.200.10"
192.168.200.1 - - [21/Jan/2020:10:26:10 +0000] "GET /favicon.ico HTTP/1.1" 404 153 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Fi
efox/72.0" "-"
[root@localhost ~]#
```

docker logs 05f56e4d8d00 -f

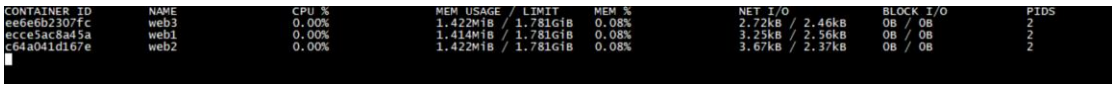
加上- f 能够实时查看新产生的日志

```
[root@localhost ~]# docker logs -f 985c03c3f3aa
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:03
          inet addr:172.17.0.3  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:656 (656.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

docker logs 命令可以查看容器的日志

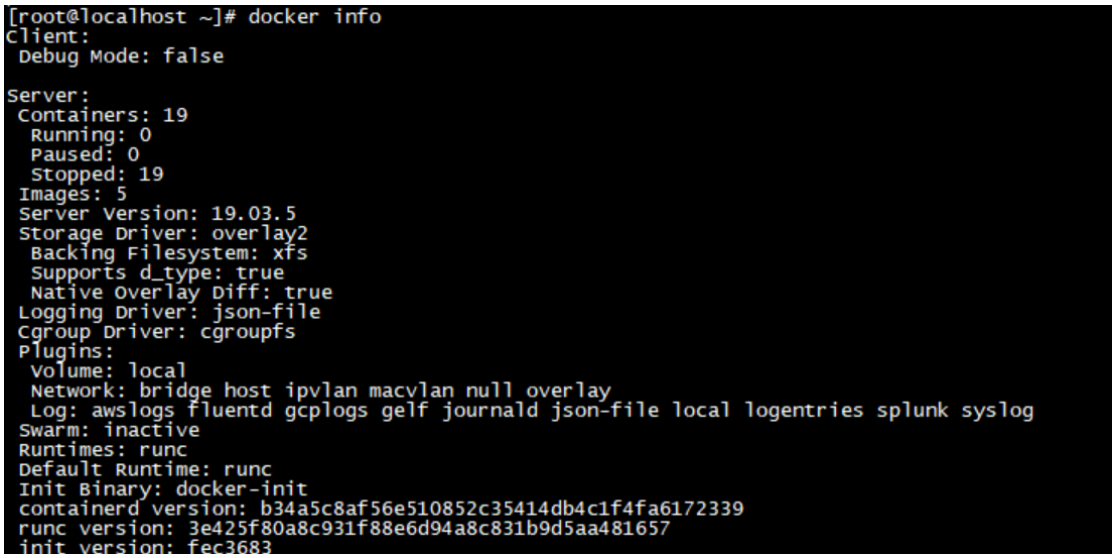
（八）查看容器资源占用

docker stats



（九）查看 docker 信息

docker info

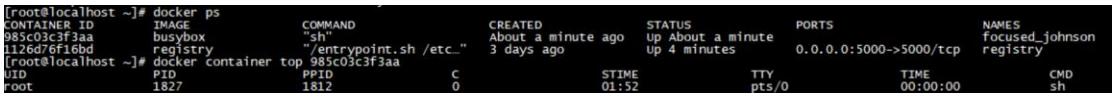


（十）查看 docker 运行的进程

docker container top 容器名称

或者

docker top 容器名称



（十一）日志管理

docker container top 容器名称

```
[root@localhost ~]# docker container top 583e4694ef7b
UID                 PID                 PPID                C                   STIME
TTY                 TIME                CMD
root                2716                2700                0                   15:48
?                   00:00:00            httpd -DFOREGROUND
bin                 2753                2716                0                   15:48
?                   00:00:00            httpd -DFOREGROUND
bin                 2754                2716                0                   15:48
?                   00:00:00            httpd -DFOREGROUND
bin                 2755                2716                0                   15:48
?                   00:00:00            httpd -DFOREGROUND
[root@localhost ~]#
```

（十二）容器中部署静态网站 nginx

1.以交互模式运行一个容器，并把容器 80 端口随机映射出来，容器运行/bin/bash，容器名称为 web

docker run -it -p 80 –name web centos /bin/bash

```
[root@localhost ~]# docker run -it -p 80 --name web centos /bin/bash
[root@37ae3be73d12 /]#
```

2.在容器内安装 nginx

yum install -y nginx

```
[root@37ae3be73d12 /]# yum install -y nginx
Failed to set locale, defaulting to C.UTF-8
CentOS-8 - AppStream            86% [=========================    ] 782 kB/s | 5.0 MB     00:01 ETA
```

3. .在容器内安装 vim

yum install -y vim

```
;@37ae3be73d12:/[root@37ae3be73d12 /]# yum install -y vim
```

4.建立网站发布目录

mkdir -p /var/www/html

```
;@37ae3be73d12:/[root@37ae3be73d12 /]# mkdir -p /var/www/html
;@37ae3be73d12:/[root@37ae3be73d12 /]# _
```

4.编辑网站网页

cd /var/www/html

vim index.html

```
;@37ae3be73d12:/[root@37ae3be73d12 /]# cd /var/www/html/
;@37ae3be73d12:/var/www/html[root@37ae3be73d12 html]# vim index.html
```

```
this is a test page
~
~
~
~
~
~
~
~
```

5.查找 nginx 安装目录

```
;@37ae3be73d12:/var/www/html[root@37ae3be73d12 html]# whereis nginx
nginx: /usr/sbin/nginx /usr/lib64/nginx /etc/nginx /usr/share/nginx /usr/share/man/man3/nginx.3pm.gz
 /usr/share/man/man8/nginx.8.gz
;@37ae3be73d12:/var/www/html[root@37ae3be73d12 html]# _
```

6.编辑配置文件

```
;@37ae3be73d12:/var/www/html[root@37ae3be73d12 html]# cd /etc/nginx/
;@37ae3be73d12:/etc/nginx[root@37ae3be73d12 nginx]# ls
conf.d                fastcgi_params          mime.types          scgi_params          win-utf
default.d             fastcgi_params.default  mime.types.default  scgi_params.default
fastcgi.conf          koi-utf                 nginx.conf          uwsgi_params
fastcgi.conf.default  koi-win                 nginx.conf.default  uwsgi_params.default
;@37ae3be73d12:/etc/nginx[root@37ae3be73d12 nginx]#
```

```
events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        server_name  _;
        root         /var/www/html;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
-- INSERT --                                                    42,31         20%
```

7.运行 nginx

nginx

```
[root@37ae3be73d12 nginx]# nginx_
```

8.查看 nginx 是否运行

ps -ef

```
;@37ae3be73d12:/etc/nginx[root@37ae3be73d12 nginx]# ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 03:41 pts/0    00:00:00 /bin/bash
root         138       1  0 04:03 ?        00:00:00 nginx: master process nginx
nginx        139     138  0 04:03 ?        00:00:00 nginx: worker process
root         140       1  0 04:04 pts/0    00:00:00 ps -ef
;@37ae3be73d12:/etc/nginx[root@37ae3be73d12 nginx]# _
```

9.ctrl+q 或者 ctrl+p 退出容器

查看容器容器还在运行

docker ps -l

```
;@37ae3be73d12:/etc/nginx[root@37ae3be73d12 nginx]# [root@localhost ~]#
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS                   NAMES
37ae3be73d12        centos              "/bin/bash"         36 minutes ago      Up 36 minutes
0.0.0.0:32768->80/tcp   web
[root@localhost ~]# _
```

10.查看容器运行的端口和容器内运行进程

docker port web

```
[root@localhost ~]# docker port web
80/tcp -> 0.0.0.0:32768
```

11.查看容器内运行进程

docker top web

```
[root@localhost ~]# docker top web
UID                 PID                 PPID                C                   STIME
TTY                 TIME                CMD
root                2418                2402                0                   11:41
?                   00:00:00            /bin/bash
root                2693                2418                0                   12:03
?                   00:00:00            nginx: master process nginx
libstor+            2694                2693                0                   12:03
?                   00:00:00            nginx: worker process
[root@localhost ~]#
```

12.访问容器内网站（使用宿主机本地地址）

```
[root@localhost ~]# curl http://127.0.0.1:32768
this is a test page
[root@localhost ~]# _
```

13.查看容器地址，使用容器地址访问网站

```
[root@localhost ~]# docker inspect web
```

```
                    "HostPort": "32768"
                }
            ]
        },
        "SandboxKey": "/var/run/docker/netns/2c6fffd05f24",
        "SecondaryIPAddresses": null,
        "SecondaryIPv6Addresses": null,
        "EndpointID": "28f22e0cd12ffb17837b4bf1fdc472e4ab031849852e3adb896985b880f6af50",
        "Gateway": "172.17.0.1",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "IPAddress": "172.17.0.3",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "MacAddress": "02:42:ac:11:00:03",
        "Networks": {
            "bridge": {
                "IPAMConfig": null,
                "Links": null,
                "Aliases": null,
                "NetworkID": "5e975fed0f5b286a2d2869d0553c48676c6994033b71baa260197561dc0dafaa",
                "EndpointID": "28f22e0cd12ffb17837b4bf1fdc472e4ab031849852e3adb896985b880f6af50"
,
                "Gateway": "172.17.0.1",
                "IPAddress": "172.17.0.3",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "MacAddress": "02:42:ac:11:00:03",
                "DriverOpts": null
            }
        }
    }
]
[root@localhost ~]# _
```

```
[root@localhost ~]# curl http://172.17.0.3
this is a test page
[root@localhost ~]# _
```

14. .访问容器内网站（使用宿主机真实地址）

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 10
00
    link/ether 00:0c:29:ab:01:f9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.100/24 brd 192.168.10.255 scope global noprefixroute ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::29f5:d088:c57a:8fa1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 10
00
    link/ether 00:0c:29:ab:01:03 brd ff:ff:ff:ff:ff:ff
    inet 192.216.5.109/24 brd 192.216.5.255 scope global noprefixroute dynamic ens37
       valid_lft 5061sec preferred_lft 5061sec
    inet6 fe80::6314:91cd:a4ea:7fe0/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:53:57:1e:76 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:53ff:fe57:1e76/64 scope link
       valid_lft forever preferred_lft forever
12: vethbfc20280@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state
UP group default
    link/ether b6:86:b9:a3:5d:43 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::b486:b9ff:fea3:5d43/64 scope link
       valid_lft forever preferred_lft forever
14: veth3c4f891@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state
UP group default
    link/ether b2:18:b0:46:4a:0b brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::b018:b0ff:fe46:4a0b/64 scope link
       valid_lft forever preferred_lft forever
[root@localhost ~]#
```

查看容器的路由

```
;@37ae3be73d12:/etc/nginx[root@37ae3be73d12 nginx]# ip ro
default via 172.17.0.1 dev eth0
172.17.0.0/16 dev eth0 proto kernel scope link src 172.17.0.3
;@37ae3be73d12:/etc/nginx[root@37ae3be73d12 nginx]# _
```

五、Docker 数据卷管理

什么是数据卷（Data Volume）

- 数据卷是经过特殊设计的目录，可以绕过联合文件系统（UFS），为一个或多个容器提供访问。

- 数据卷设计的目的，在于数据的永久化，它完全独立与容器的生存周期，因此，Docker不会在容器删除时删除其挂载的数据卷，也不会存在类似的垃圾收集机制，对容器引用的数据卷进行处理。

（一）随机生成指定挂载目录

1.运行一个容器，-h 是指定主机名，-v 指定挂载数据位置

docker run -it –name data -v /data -h datatest centos

```
[root@localhost ~]# docker run -it --name data -v /data -h datatest centos
[root@datatest /]#
```

data 文件夹原来 centos 里面是没有的

```
[root@datatest /]# ls
bin   dev  home  lib64       media  opt   root  sbin  sys  usr
data  etc  lib   lost+found  mnt    proc  run   srv   tmp  var
;@datatest:/[root@datatest /]# _
```

data 文件夹内是空的

```
;@datatest:/[root@datatest /]# ls /data
;@datatest:/[root@datatest /]#
```

2.在宿主机上检查实际挂载的位置上

docker inspect -f "{{.Mounts}}" data data 是容器名称

```
[root@localhost ~]# docker inspect -f "{{.Mounts}}" data
[{volume d34a6651392ad5dca4ab3248243efe2090006dd1da9fbd1a095b2b4909b1724d /var/lib/docker/volumes/d34a6651392ad5dca4ab3248243efe2090006dd1da9fbd1a095b2b4909b1724d/_data /data local  true }]
[root@localhost ~]#
```

```
[root@localhost ~]# cd /var/lib/docker/volumes/d34a6651392ad5dca4ab3248243efe2090006dd1da9fbd1a095b2b4909b1724d/
[root@localhost d34a6651392ad5dca4ab3248243efe2090006dd1da9fbd1a095b2b4909b1724d]# ls
_data
[root@localhost d34a6651392ad5dca4ab3248243efe2090006dd1da9fbd1a095b2b4909b1724d]# cd _data/
[root@localhost _data]# ls
[root@localhost _data]# touch ceshi
[root@localhost _data]#
```

容器内操作

```
;@datatest:/[root@datatest /]# ls /data
ceshi
;@datatest:/[root@datatest /]#
```

```
;@datatest:/[root@datatest /]# cd /data
;@datatest:/data[root@datatest data]# mkdir test
;@datatest:/data[root@datatest data]# ls
ceshi  test
;@datatest:/data[root@datatest data]# _
```

在宿主机上操作

```
[root@localhost _data]# touch ceshi
[root@localhost _data]# ls
ceshi  test
[root@localhost _data]#
```

（二）指定挂载目录

1.运行一个容器，-h 是指定主机名，-v 指定挂载数据位置

docker run -it –name data2 -v /opt:/opt -h datatest centos

```
[root@localhost ~]# docker run -it --name data2  -h datatest -v /opt:/opt centos
[root@datatest /]# ls
bin  etc   lib    lost+found  mnt  proc  run   srv  tmp  var
dev  home  lib64  media       opt  root  sbin  sys  usr
;@datatest:/[root@datatest /]# _
```

docker run -it –name data 3-v /opt:/opt:ro -h datatest centos

加上 ro，只读不能写，默认读写

```
[root@localhost _data]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
[root@localhost _data]#
```

（三）数据卷容器

将运行容器作为数据卷

docker run -it –name data 4 --volumes-from data centos,data 这个容器是上面实验运行过的，运行还是没有运行都可以

```
[root@localhost ~]# docker run -it --name data4  --volumes-from data centos
[root@dcf1e3c5c1b7 /]# ls
bin   dev  home  lib64       media  opt   root  sbin  sys  usr
data  etc  lib   lost+found  mnt    proc  run   srv   tmp  var
;@dcf1e3c5c1b7:/[root@dcf1e3c5c1b7 /]# ls /data/
ceshi  test
;@dcf1e3c5c1b7:/[root@dcf1e3c5c1b7 /]#
```

（四）数据卷备份与还原

1.创建数据卷容器 db

docker run -it -v /db:/db --name db centos

2.创建 db1 和 db2 两个容器，并使用--volumes-from 挂载 dbdata 容器中的数据卷

docker run -it --volumes-from db --name db1 centos

docker run -it --volumes-from db --name db2 centos

这样在三个容器中任何一个操作，在其他的容器都是可以看得见的，如果我们把数据卷容器停止或者删除掉，挂载数据卷容器的容器依然可以使用数据卷，也就是说数据卷容器在

```
[root@localhost ~]# docker run -it -v /db:/db --name db centos
[root@a0c2aea42b7e /]# [root@localhost ~]#
[root@localhost ~]# dockr ps -l
-bash: dockr: command not found
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
a0c2aea42b7e        centos              "/bin/bash"         35 seconds ago      Up 32 seconds                           db
[root@localhost ~]# docker run -it --volumes-from db --name db1 centos
[root@b34058bcf752 /]#
[root@b34058bcf752 /]# [root@localhost ~]#
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
b34058bcf752        centos              "/bin/bash"         28 seconds ago      Up 27 seconds                           db1
[root@localhost ~]# docker run -it --volumes-from db --name db2 centos
[root@b75f0e7d15fa /]# [root@localhost ~]#
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
b75f0e7d15fa        centos              "/bin/bash"         17 seconds ago      Up 15 seconds                           db2
[root@localhost ~]#
```

3.备份

备份：docker run --volumes-from db -v /backup1:/backup2:wr --name=db3 centos tar cvf /backup/db3.tar /datavolume1（

这个容器的名字叫做 db3，挂载在 db 上，宿主机目录是 backup1，容器中的目录是 backup2，权限是读写权限，运行 ubuntu 系统，压缩之后的路径是 /backup/db3.tar，/datavolume1 是需要压缩备份的目录）：

将一个包含数据卷的容器中的数据，通过一个容器执行一个压缩命令，从而将数据备份出来。

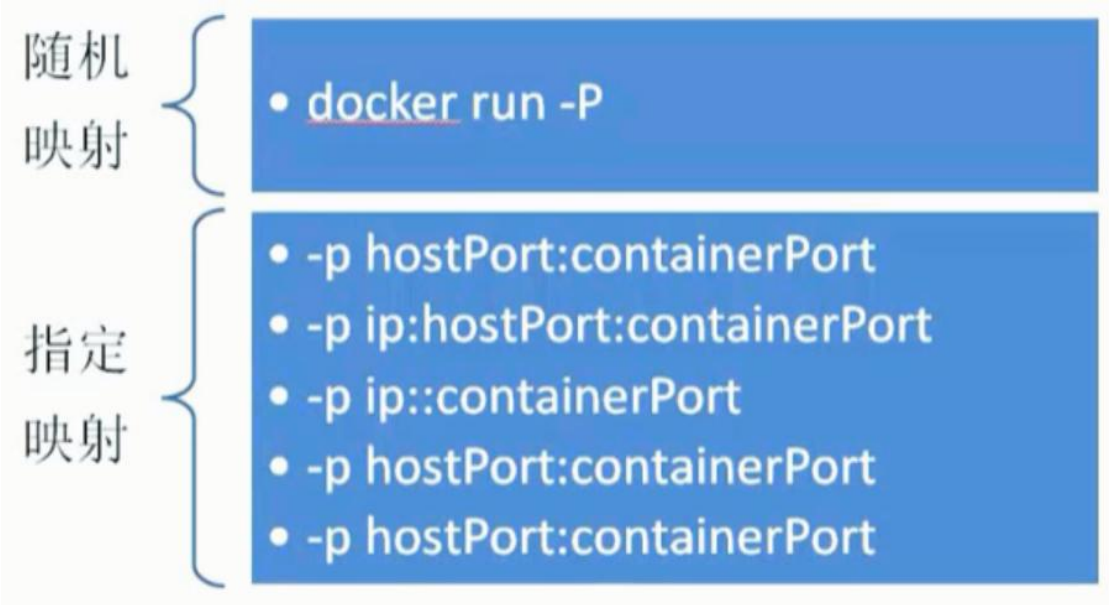
4.还原

$docker run --volumes-from db -v /huanyuan:/huanyuan centos tar xvf /backup/db3.tar /datavolume1

六、Docker 端口映射

（一）随机映射

大写 P，随机映射一个端口



docker run -d -P --name myhttpd httpd

```
[root@localhost ~]# docker run -d -P --name myhttpd httpd
2660c3b0a07bab27e4a91bd1735411174bfc61c93b62c4f71279c26ea666527d
[root@localhost ~]#
```

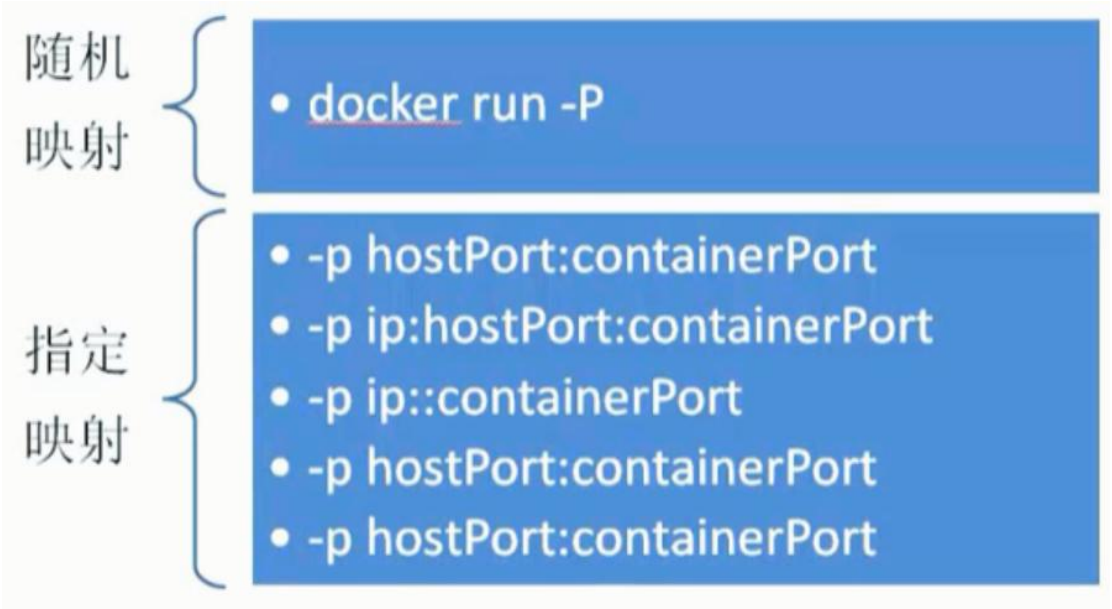
docker ps -l

```
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS                     NAM
ES
2660c3b0a07b        httpd               "httpd-foreground"   About a minute ago  Up About a minute   0.0.0.0:32768->80/tcp     myh
ttpd
[root@localhost ~]#
```

验证访问

（二）端口映射（指定端口）

大写 P，随机映射一个端口



小写 p，指定端口

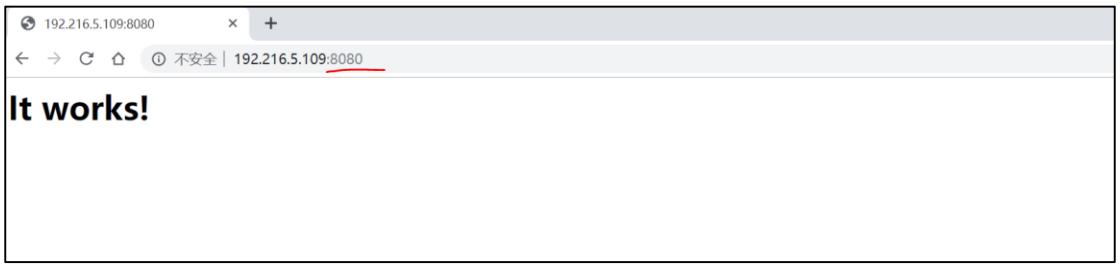
docker run -d -p 8080:80 --name myhttpd2 httpd

```
[root@localhost ~]# docker run -d -p 8080:80 --name myhttpd2 httpd
006b0f8c56696b03324279912d2a82f76060c0921b94925498c5006b3e36478c
[root@localhost ~]#
```

docker ps -l

```
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND              CREATED             STATUS              PORTS                     NAMES
006b0f8c5669        httpd               "httpd-foreground"   About a minute ago  Up About a minute   0.0.0.0:8080->80/tcp      myhttpd2
[root@localhost ~]#
```

验证访问



brctl show

```
[root@localhost ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.02429853bf39       no              veth000e740
                                                        vethdfa266e
[root@localhost ~]#
```

iptables -t nat -L -n

```
[root@localhost ~]# iptables -t nat -L -n
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  0.0.0.0/0            0.0.0.0/0            ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  0.0.0.0/0           !127.0.0.0/8          ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.17.0.0/16        0.0.0.0/0
MASQUERADE  tcp  --  172.17.0.2           172.17.0.2           tcp dpt:80
MASQUERADE  tcp  --  172.17.0.3           172.17.0.3           tcp dpt:80

Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:32768 to:172.17.0.2:80
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:8080 to:172.17.0.3:80
[root@localhost ~]#
```

七、Docker compose

Compose 是用于定义和运行多容器 Docker 应用程序的工具。通过 Compose，您可以使用 YML 文件来配置应用程序需要的所有服务。然后，使用一个命令，就可以从 YML 文件配置中创建并启动所有服务。

使用 yaml 注意事项：

（1）使用空格进行缩进

（2）通常开头缩进两个空格

（3）在 yaml 里，用#做注释

（4）要在字符（冒号、逗号、横杠）后缩进一个空格

（5）如果包含特殊字符要用单引号引起来

（6）布尔值必须用双引号引起来

Compose 使用的三个步骤：

- 使用 Dockerfile 定义应用程序的环境。

- 使用 docker-compose.yml 定义构成应用程序的服务，这样它们可以在隔离环境中一起运行。
- 最后，执行 docker-compose up 命令来启动并运行整个应用程序。

（一）docker compose 安装下载

1.下载 Compose

下载地址

https://github.com/docker/compose/releases

方法 1

curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

uname -s 显示操作系统名称

uname -m 显示操作系统 32 位还是 64 位

下载 docker-compose 保存到/usr/local/bin/

要安装其他版本的 Compose，请替换 1.24.1。

```
[root@centos71611 ~]#  curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-c
ompose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   651  100   651    0     0    431      0  0:00:01  0:00:01 --:--:--   431
100 15.4M  100 15.4M    0     0   347k      0  0:00:45  0:00:45 --:--:--  501k
[root@centos71611 ~]#
```

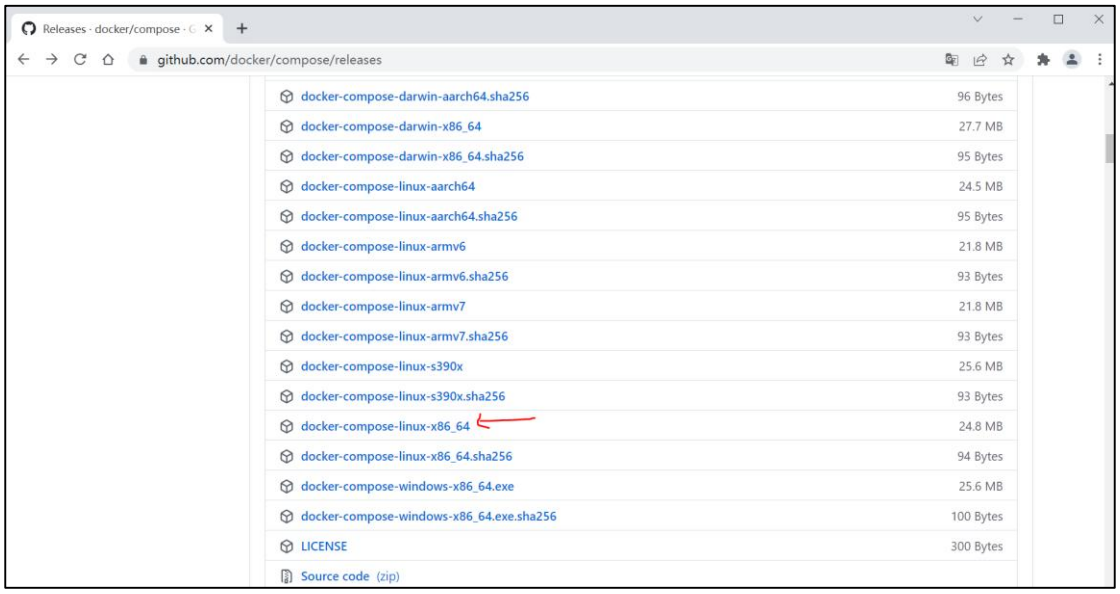
ls /usr/local/bin/

```
[root@centos71611 ~]# ls  /usr/local/bin/
docker-compose
[root@centos71611 ~]#
```

方法 2

（1）官网直接下载

https://github.com/docker/compose/releases



（1）将 docker-cmpose 上传到/root

（2）移动到/usr/local/bin

cd ~

mv docker-compose-linux-x86_64 /usr/local/bin/docker-compose

2.将可执行权限应用于二进制文件

chmod +x /usr/local/bin/docker-compose

```
[root@centos71611 ~]# chmod +x /usr/local/bin/docker-compose
[root@centos71611 ~]#
```

docker-compose --version

查看测试是否安装成功

```
[root@centos71611 ~]# docker-compose --version
docker-compose version 1.24.1, build 4667896b
[root@centos71611 ~]#
```

（二）发布三个 nginx 容器

1.编辑 docker-compose.yml 文件

Compose 允许用户通过一个 docker-compose.yml 模板文件（YAML 格式）来定义一组相关联的应用容器为一个项目（project）。

Compose 模板文件是一个定义服务、网络和卷的 YAML 文件。Compose 模板文件默认路径是当前目录下的 docker-compose.yml，可以使用.yml 或.yaml 作为文件扩展名。

Docker-Compose 标准模板文件应该包含 version、services、networks 三大部分，最关键的是 services 和 networks 两个部分。

vi docker-compose.yml

docker-compose.yml 文件指定了 3 个 web 服务

```
version: '3'
services:
  web1:
    image: nginx
    ports:
      - "6061:80"
    container_name: "web1"
    networks:
      - dev
  web2:
    image: nginx
    ports:
      - "6062:80"
    container_name: "web2"
    networks:
      - dev
      - pro
  web3:
    image: nginx
    ports:
      - "6063:80"
    container_name: "web3"
    networks:
      - pro

networks:
  dev:
    driver: bridge
  pro:
    driver: bridge
~
~
```

```
version: '3'
services:
    web1:
        image: nginx
        ports:
            - "6061:80"
        container_name: "web1"
        networks:
            - dev
    web2:
        image: nginx
        ports:
            - "6062:80"
        container_name: "web2"
        networks:
            - dev
            - pro
    web3:
        image: nginx
```

```
        ports:
            - "6063:80"
        container_name: "web3"
        networks:
            - pro

networks:
    dev:
        driver: bridge
    pro:
        driver: bridge
```

2. 启动应用

docker-compose up -d

在当前目录执行不用输入文件 yml 名称

-d 后台启动并运行所有的容器

```
[root@master ~]# docker-compose up -d
Creating network "root_dev" with driver "bridge"
Creating network "root_pro" with driver "bridge"
Pulling web1 (nginx:)...
latest: Pulling from library/nginx
852e50cd189d: Pull complete
571d7e852307: Pull complete
addb10abd9cb: Pull complete
d20aa7ccdb77: Pull complete
8b03f1e11359: Pull complete
Digest: sha256:6b1daa9462046581ac15be20277a7c75476283f969cb3a61c8725ec38d3b01c3
Status: Downloaded newer image for nginx:latest
Creating web3 ... done
Creating web1 ... done
Creating web2 ... done
[root@master ~]#
```

3.访问服务

http://192.216.5.210:6061/

192.216.5.210 是宿主机的 ip 地址

http://192.216.5.210:6062/

http://192.216.5.210:6063/

（三）发布 mysql 容器

1.编辑 docker-compose.yml 文件


```
version: '3.7'
services:
   mysql:
      image: mysql:8
      container_name: mysql8
      ports:
      - 3306:3306
      command:
         --default-authentication-plugin=mysql_native_password
         --character-set-server=utf8mb4
         --collation-server=utf8mb4_general_ci
         --explicit_defaults_for_timestamp=true
         --lower_case_table_names=1
      environment:
      - MYSQL_ROOT_PASSWORD=root
      volumes:
      - /etc/localtime:/etc/localtime:ro
      - volumes.mysql8-data:/var/lib/mysql
volumes:
   volumes.mysql8-data: null
```

2. 启动应用

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

```
[root@centos7test ~]#  docker-compose up -d
[+] Running 1/1
 ⠿ mysql Pulled                                                                                                                                                  4.9s
WARN[0005] Found orphan containers ([web2 web1 web3]) for this project. If you removed or renamed this service in your compose file, you can run this command
 with the --remove-orphans flag to clean it up.
[+] Running 6/6
 ⠿ Network root_default                  Created                                                                                                                 0.1s
 ⠿ Volume "root_volumes.mysql8-data"     Created                                                                                                                 0.0s
 ⠿ Container mysql8                      Started                                                                                                                 3.1s
 ⠿ Container web2                        Started                                                                                                                 3.1s
 ⠿ Container web3                        Started                                                                                                                 2.9s
 ⠿ Container web1                        Started                                                                                                                 3.1s
[root@centos7test ~]# netstat -pantu | grep 3306
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN      3335/docker-proxy
tcp6       0      0 :::3306                 :::*                    LISTEN      3341/docker-proxy
```

3.访问 mysql 服务

mysql -h 127.0.0.1 -u root -p

密码是 root

```
[root@centos7test ~]# mysql -h 127.0.0.1 -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

（四）发布 wordpress 容器

1.编辑 docker-compose.yml 文件

```
[root@centos7test ~]# vi docker-compose.yml

version: '3.3'
services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress
   wordpress:
     depends_on:
       - db
     image: wordpress:5.6-php7.3-apache
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}
```

```
version: '3.3'
services:
    db:
      image: mysql:5.7
      volumes:
        - db_data:/var/lib/mysql
      restart: always
      environment:
        MYSQL_ROOT_PASSWORD: somewordpress
        MYSQL_DATABASE: wordpress
        MYSQL_USER: wordpress
        MYSQL_PASSWORD: wordpress
    wordpress:
      depends_on:
        - db
      image: wordpress:5.6-php7.3-apache
      ports:
        - "8000:80"
      restart: always
      environment:
```

```
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}
```

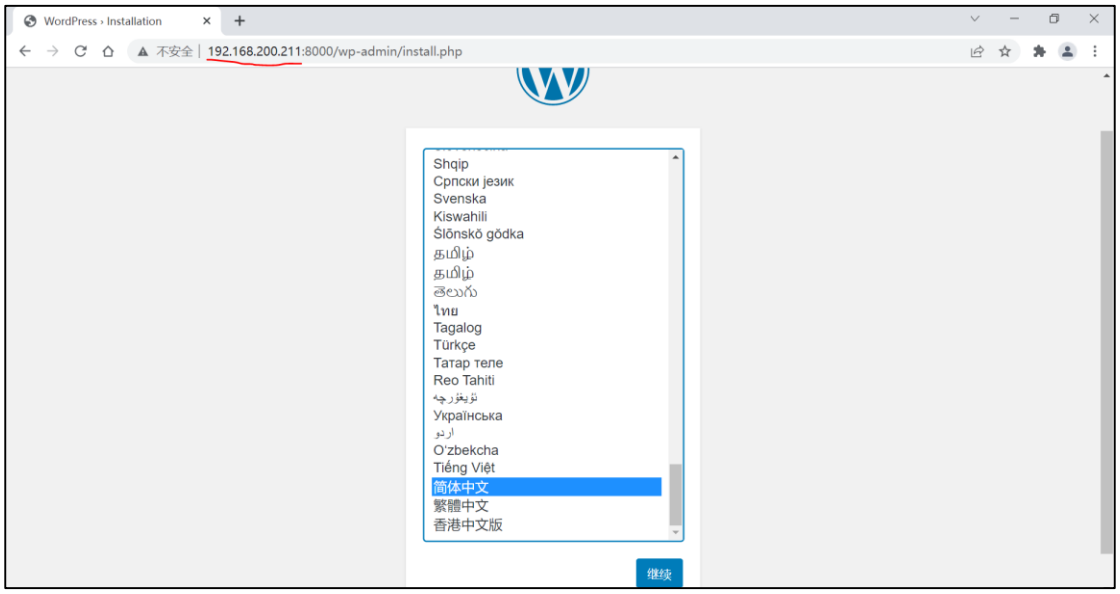
2. 启动应用

docker-compose up -d

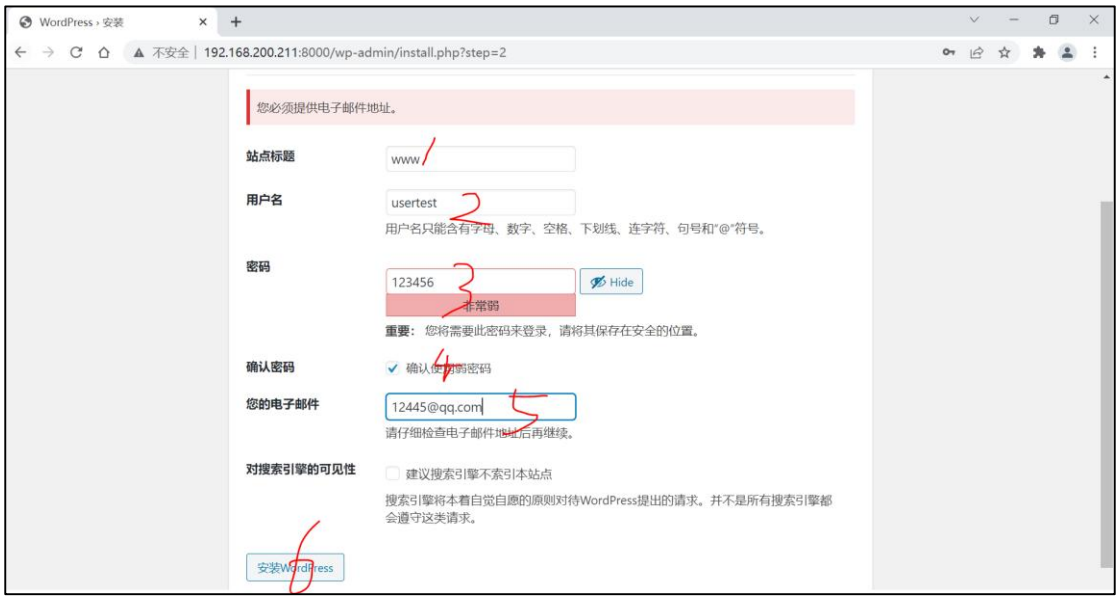
```
[root@centos7test ~]# docker-compose up -d
WARN[0000] Found orphan containers ([mysql8_web2_web1_web3]) for this project. If you removed or renamed this service in your compose file, you can run this
command with the --remove-orphans flag to clean it up.
[+] Running 3/3
 ⠿ Volume "root_db_data"       Created                                                                                                              0.0s
 ⠿ Container root-db-1         Started                                                                                                              1.2s
 ⠿ Container root-wordpress-1  Started                                                                                                              3.4s
[root@centos7test ~]# netstat -pantu | grep 8000
tcp        0      0 0.0.0.0:8000            0.0.0.0:*               LISTEN      9654/docker-proxy
tcp6       0      0 :::8000                 :::*                    LISTEN      9659/docker-proxy
```

3.访问

（1）http://192.168.200.211:8000

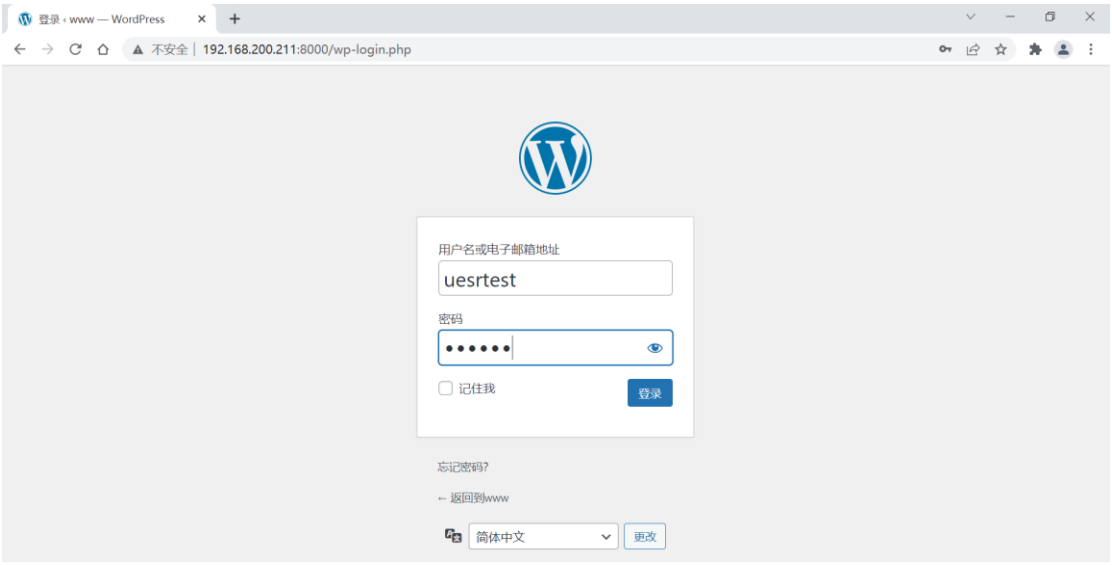


（2）填写信息



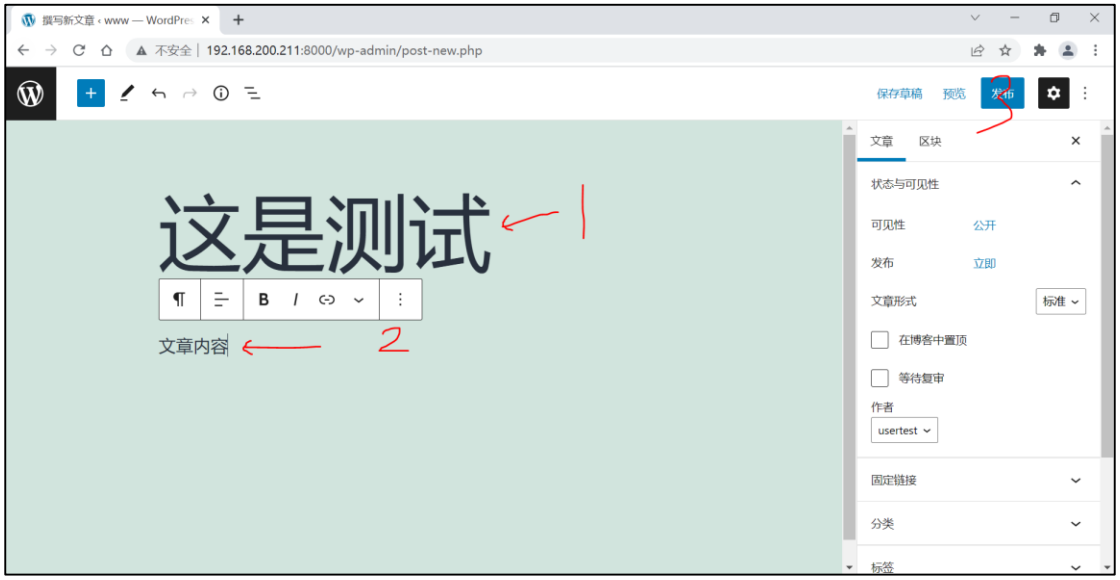
（3）初始化完成，登录

（4）登录后台



（5）进入到后台

（6）



（五）发布 postgres 容器

1.编辑 docker-compose.yml 文件

vi docker-compose.yml

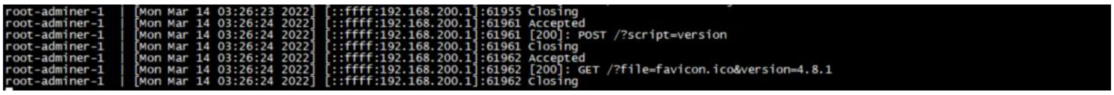
```
[root@centos7test ~]# vi docker-compose.yml
version: '3.1'

services:

  db:
    image: postgres
    restart: always
    environment:
      POSTGRES_PASSWORD: example

  adminer:
    image: adminer
    restart: always
    ports:
      - 8080:8080
~
```

```
version: '3.1'
```

```
services:
  db:
    image: postgres
    restart: always
    environment:
      POSTGRES_PASSWORD: example
  adminer:
    image: adminer
    restart: always
    ports:
      - 8080:8080
```

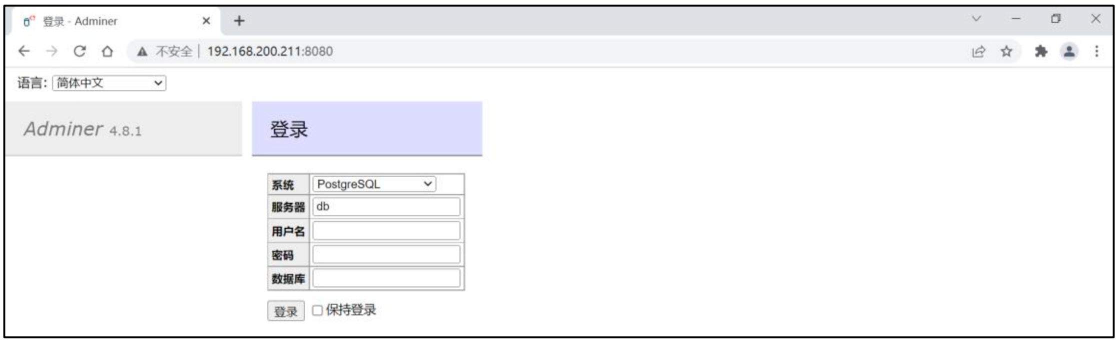
2. 运行

docker-compose up



3.访问

http://192.168.200.211:8080/



（六）发布 Django 容器 1

目标：

使用 Docker Compose 建立和运行一个简单的 Django/PostgreSQL 应用程序。

实施思路：

项目中需要创建一个 Dockerfile 文件、一个 Python 依赖文件和一个名为 docker- compose.yml 的 Compose 文件。

1.编辑# vi docker-compose.yml 文件

vi Dockerfile

```
[root@centos7test ~]# vi Dockerfile

# 从Python 3父镜像开始
FROM python:3
ENV PYTHONUNBUFFERED 1
# 在镜像中添加code目录
RUN mkdir /code
WORKDIR /code
COPY requirements.txt /code/
# 在镜像中安装由requirements.txt文件指定要安装的Python依赖
RUN pip install -r requirements.txt
COPY . /code/
```

```
# 从 Python 3 父镜像开始
FROM python:3
ENV PYTHONUNBUFFERED 1
# 在镜像中添加 code 目录
RUN mkdir /code
WORKDIR /code
COPY requirements.txt /code/
# 在镜像中安装由 requirements.txt 文件指定要安装的 Python 依赖
RUN pip install -r requirements.txt
COPY . /code/
```

2.编辑 docker-compose.yml 文件

vi docker-compose.yml

文件将把所有的东西关联起来。它描述了应用的构成（一个 web 服务和一个数据库）、使用的 Docker 镜像、镜像之间的连接、挂载到容器的卷，以及服务开放的端口。

```
[root@centos7test django]# vi docker-compose.yml

version: "3"
services:
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
```

```
version: "3"
services:
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
```

3.编辑 requirements.txt 文件

vi requirements.txt

文件里面写明需要安装的具体依赖包名和版本限定

```
[root@centos7test ~]# vi requirements.txt

Django>=2.0,<3.0
psycopg2>=2.7,<3.0
```

```
Django>=2.0,<3.0
psycopg2>=2.7,<3.0
```

4.创建 Django 项目

docker-compose run web django-admin startproject django_example .

为 web 服务构建一个镜像，接着使用这个镜像在容器里运行 django-admin startproject django_example 指令。

```
[root@centos7test django]# docker-compose run web django-admin startproject django_example .
[+] Running 1/0
 ∷ Network django_default  Created                                                          0.0s
Sending build context to Docker daemon     399B
Step 1/7 : FROM python:3
 ---> 178dcaa62b39
Step 2/7 : ENV PYTHONUNBUFFERED 1
 ---> Running in 6b33be644898
Removing intermediate container 6b33be644898
 ---> a53a9bac2d65
Step 3/7 : RUN mkdir /code
 ---> Running in d6fb4dee817e
Removing intermediate container d6fb4dee817e
 ---> 12d5d8f6bf80
Step 4/7 : WORKDIR /code
 ---> Running in ecc8aafc4e4c
Removing intermediate container ecc8aafc4e4c
 ---> 8ec9c3ae2654
Step 5/7 : COPY requirements.txt /code/
```

```
Successfully built psycopg2
Installing collected packages: sqlparse, pytz, psycopg2, Django
Successfully installed Django-2.2.27 psycopg2-2.9.3 pytz-2021.3 sqlparse-0.4.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use
a virtual environment instead: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.2.4; however, version 22.0.4 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.
Removing intermediate container 9ec91bd1e0d2
 ---> 6b8b8d79fcee
Step 7/7 : COPY . /code/
 ---> d9edc4f765b5
Successfully built d9edc4f765b5
Successfully tagged root_web:latest
[root@centos7test ~]#
```

ls -l

```
[root@centos7test django]# ls -l
total 16
drwxr-xr-x 2 root root  74 Mar 14 05:33 django_example
-rw-r--r-- 1 root root 156 Mar 14 05:32 docker-compose.yml
-rw-r--r-- 1 root root 254 Mar 14 05:30 Dockerfile
-rwxr-xr-x 1 root root 634 Mar 14 05:33 manage.py
-rw-r--r-- 1 root root  37 Mar 14 02:13 requirements.txt
[root@centos7test django]#
```

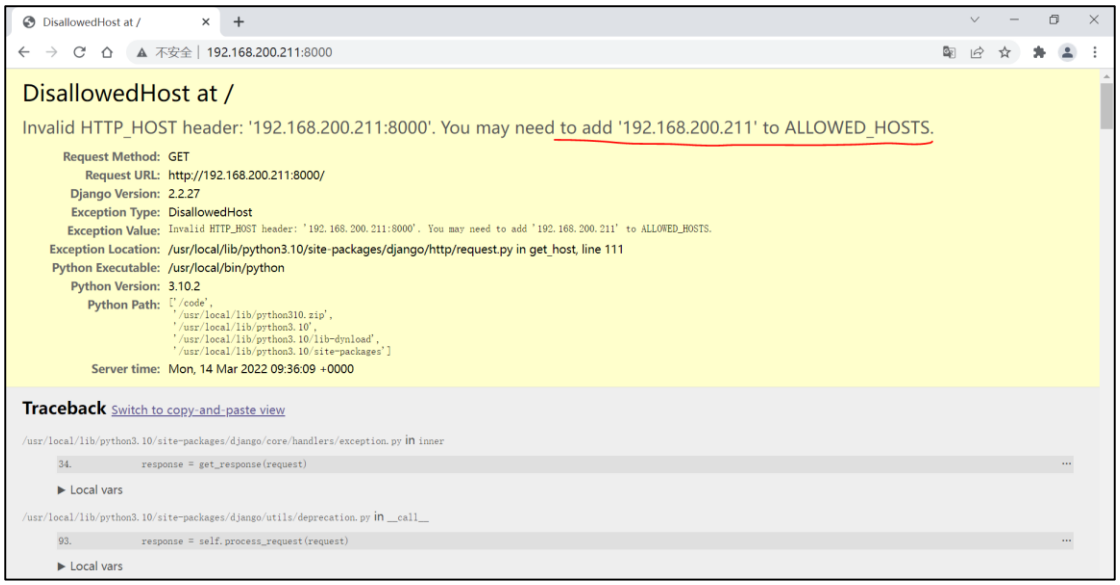
5.编辑项目目录中的 django_example/settings.py 文件

vi django_example/settings.py

为应用设置好访问地址

```
[root@centos7test django]# vi django_example/settings.py
"""
Django settings for django_example project.

Generated by 'django-admin startproject' using Django 2.2.27.

For more information on this file, see
https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.2/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'y)n24+-+59%rh9zc8j_gb67l1!q=c4djda00$(a#$tmy=+m2$l'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ["192.168.200.211"]


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
```

否则

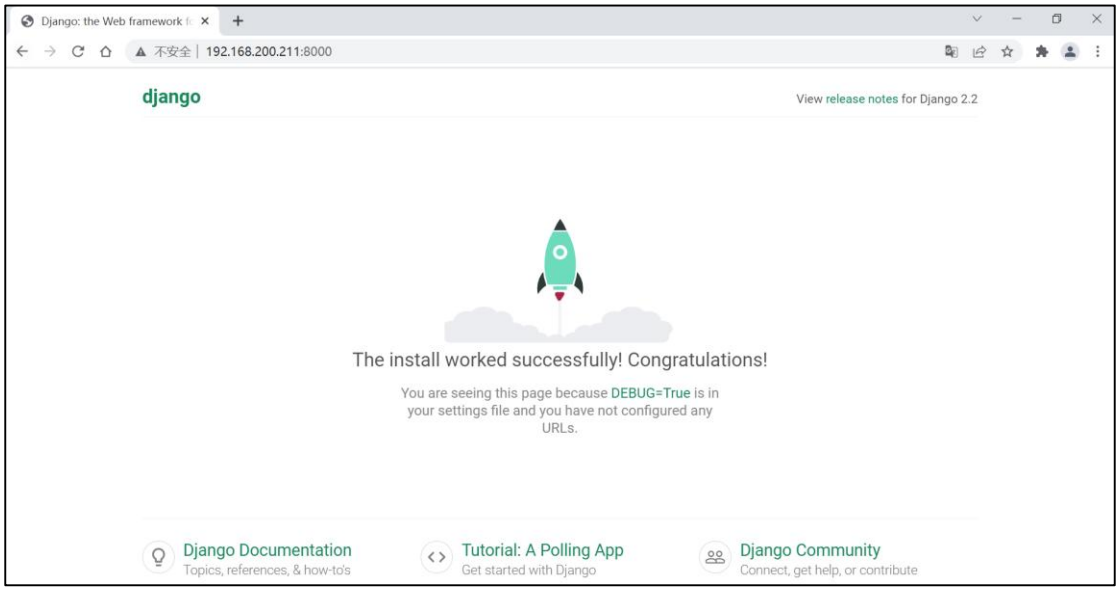


6.在项目目录的根目录下执行 docker-compose up 命令

docker-compose up

7.访问

http://192.168.200.211:8000/



docker-compose up -d db

docker-compose up -d web

（七）发布 Django 容器 2

1.编辑# vi docker-compose.yml 文件

vi Dockerfile

```
[root@centos7test ~]# vi Dockerfile

# 从Python 3父镜像开始
FROM python:3
ENV PYTHONUNBUFFERED 1
# 在镜像中添加code目录
RUN mkdir /code
WORKDIR /code
COPY requirements.txt /code/
# 在镜像中安装由requirements.txt文件指定要安装的Python依赖
RUN pip install -r requirements.txt
COPY . /code/
```

```
# 从 Python 3 父镜像开始
FROM python:3
ENV PYTHONUNBUFFERED 1
# 在镜像中添加 code 目录
RUN mkdir /code
WORKDIR /code
COPY requirements.txt /code/
# 在镜像中安装由 requirements.txt 文件指定要安装的 Python 依赖
RUN pip install -r requirements.txt
COPY . /code/
```

2.编辑 docker-compose.yml 文件

vi docker-compose.yml

```
[root@centos7test bowen]# vi docker-compose.yml

version: '3'
services:
  db:
    image: postgres:9.6.23
    volumes:
      - db_data:/var/lib/postgresql
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - db
volumes:
    db_data: {}
```

```
version: '3'
services:
   db:
      image: postgres:9.6.23
      volumes:
         - db_data:/var/lib/postgresql
   web:
      build: .
      command: python manage.py runserver 0.0.0.0:8000
      volumes:
         - .:/code
      ports:
         - "8000:8000"
      depends_on:
         - db
volumes:
```

db_data: {}

3.编辑 requirements.txt 文件

vi requirements.txt

文件里面写明需要安装的具体依赖包名和版本限定

```
[root@centos7test ~]# vi requirements.txt

Django>=2.0,<3.0
psycopg2>=2.7,<3.0
```

Django>=2.0,<3.0

psycopg2>=2.7,<3.0

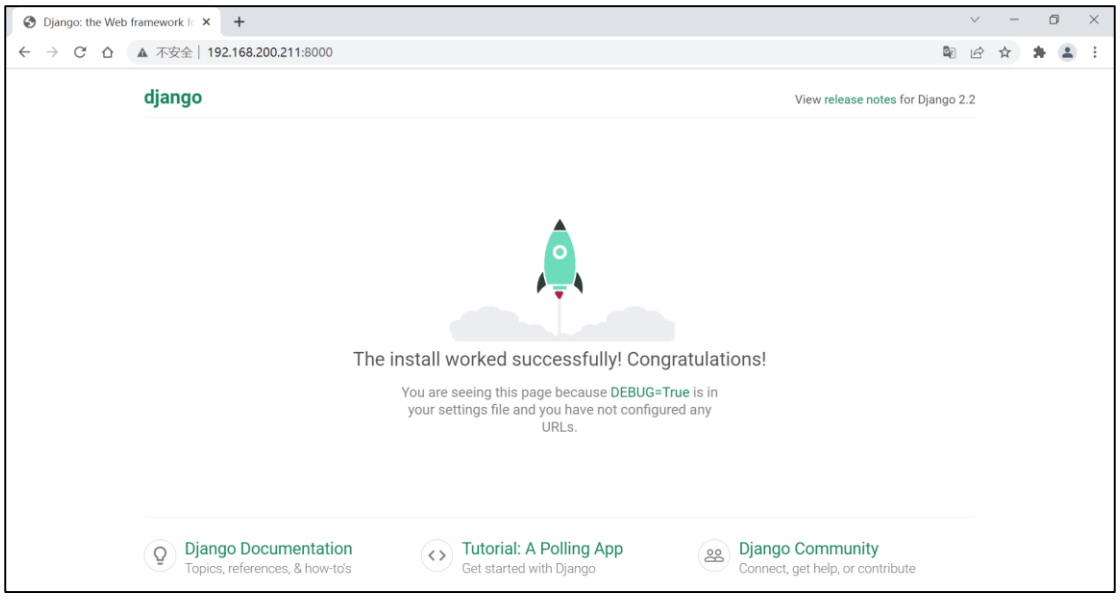
4.创建 Django 项目

docker-compose run web django-admin startproject django_example .

docker-compose up

```
[root@centos7test bowen]# docker-compose up
[+] Running 2/0
 # Container bowen-db-1   Created                                                                                    0.0s
 # Container bowen-web-1  Created                                                                                    0.0s
Attaching to bowen-db-1, bowen-web-1
bowen-db-1    | Error: Database is uninitialized and superuser password is not specified.
bowen-db-1    |        You must specify POSTGRES_PASSWORD to a non-empty value for the
bowen-db-1    |        superuser. For example, "-e POSTGRES_PASSWORD=password" on "docker run".
bowen-db-1    |
bowen-db-1    |        You may also use "POSTGRES_HOST_AUTH_METHOD=trust" to allow all
bowen-db-1    |        connections without a password. This is *not* recommended.
bowen-db-1    |
bowen-db-1    |        See PostgreSQL documentation about "trust":
bowen-db-1    |        https://www.postgresql.org/docs/current/auth-trust.html
bowen-db-1 exited with code 1
bowen-web-1   | Watching for file changes with StatReloader
bowen-web-1   | Performing system checks...
bowen-web-1   |
bowen-web-1   | System check identified no issues (0 silenced).
bowen-web-1   |
bowen-web-1   | You have 17 unapplied migration(s). Your project may not work properly until you apply the migrations for ap
p(s): admin, auth, contenttypes, sessions.
bowen-web-1   | Run 'python manage.py migrate' to apply them.
bowen-web-1   | March 15, 2022 - 03:16:31
bowen-web-1   | Django version 2.2.27, using settings 'myexample.settings'
bowen-web-1   | Starting development server at http://0.0.0.0:8000/
bowen-web-1   | Quit the server with CONTROL-C.
```

http://192.168.200.211:8000/



八、Docker 网络

Docker 网络

安装 docker，它会自动创建三个网络

docker network ls

```
[root@centos71611 ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
5aca57024c7e        bridge              bridge              local
52b2e966a632        host                host                local
b17e9dae02ee        none                null                local
[root@centos71611 ~]#
```

（一）none 网络

docker run -it --network=none busybox

```
[root@localhost ~]# docker run -it --network=none busybox
/ # ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ #
```

（二）host 网络

docker run -it --network=host busybox

```
[root@localhost ~]# docker run -it --network=host busybox
/ # ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:7A:22:74:44
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:7aff:fe22:7444/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:10252 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14633 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:422415 (412.5 KiB)  TX bytes:37965218 (36.2 MiB)

ens33     Link encap:Ethernet  HWaddr 00:0C:29:D1:51:FC
          inet addr:192.168.200.10  Bcast:192.168.200.255  Mask:255.255.255.0
          inet6 addr: fe80::29f5:d088:c57a:8fa1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:256271 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37602 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:368084872 (351.0 MiB)  TX bytes:3154768 (3.0 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:75 errors:0 dropped:0 overruns:0 frame:0
          TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7744 (7.5 KiB)  TX bytes:7744 (7.5 KiB)

/ #
```

（三）bridge 网络

容器默认都会挂载此网络

1.# brctl show

```
[root@localhost ~]# brctl show
-bash: brctl: command not found
```

2.安装 brctl

yum install -y bridge-utils

```
[root@localhost ~]# yum install -y bridge-utils
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.bit.edu.cn
 * extras: mirrors.huaweicloud.com
 * updates: mirrors.huaweicloud.com
```

3.# brctl show

目前还没有容器运行

```
[root@localhost ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.02427a227444       no
[root@localhost ~]#
```

4. 运行一个容器

docker run -it -d centos

```
[root@localhost ~]# docker run  -it -d  centos
093a27ec55e6e611778f2a3c7de570c39aa1ec2fe9c7a01d0106edc6eb3d1fd0
```

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
093a27ec55e6        centos              "/bin/bash"         2 seconds ago       Up 1 second                             practical_blackwell
```

docker exec -it 093a27ec55e6 bash

```
[root@localhost ~]# docker exec -it 093a27ec55e6  bash
[root@093a27ec55e6 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
32: eth0@if33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
[root@093a27ec55e6 /]#
```

5.# brctl show

```
[root@localhost ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.02427a227444       no              veth083ad51
```

docker run -it busybox

```
[root@localhost ~]# docker run -it  busybox
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:516 (516.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ #
```

查看本地桥接网络配置

docker network inspect bridge

```
[root@localhost ~]# docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "6878fd890d57e00affa179ef7fb2dc3d1bb58b508b099a6aafc97850de7255d5",
        "Created": "2020-01-20T18:23:37.841174986+08:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "c560b74035daad6a1fae73dfc6960a4c32e46ce9c2891998056bb8ee6d13bdd3": {
                "Name": "cranky_vaughan",
                "EndpointID": "b2726199ac39d07f0ece40eb0b93ffddefbbb3b106d2c7da978acbe5ef40e208",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
```

ifconfig

```
[root@localhost dockerfile]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:abff:fe96:3fb0  prefixlen 64  scopeid 0x20<link>
        ether 02:42:ab:96:3f:b0  txqueuelen 0  (Ethernet)
        RX packets 3299  bytes 164008 (160.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3713  bytes 24069602 (22.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.100  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::29f5:d088:c57a:8fa1  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:ab:01:f9  txqueuelen 1000  (Ethernet)
        RX packets 17716  bytes 24880744 (23.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3944  bytes 267796 (261.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ifconfig docker0 192.168.168.1 netmask 255.255.255.0

```
[root@localhost dockerfile]# ifconfig docker0 192.168.168.1 netmask 255.255.255.0
[root@localhost dockerfile]#
```

ifconfig

```
[root@localhost dockerfile]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.168.1  netmask 255.255.255.0  broadcast 192.168.168.255
        inet6 fe80::42:abff:fe96:3fb0  prefixlen 64  scopeid 0x20<link>
        ether 02:42:ab:96:3f:b0  txqueuelen 0  (Ethernet)
        RX packets 3299  bytes 164008 (160.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3713  bytes 24069602 (22.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.100  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::29f5:d088:c57a:8fa1  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:ab:01:f9  txqueuelen 1000  (Ethernet)
        RX packets 17723  bytes 24881284 (23.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3952  bytes 268426 (262.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

systemctl restart docker

```
[root@localhost dockerfile]# docker run -it -d centos
ad0436d6460cde10e660f29d896de6dd589038287697f77def68bd66700f5d87
[root@localhost dockerfile]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
ad0436d6460c        centos              "/bin/bash"         9 seconds ago       Up 8 seconds
[root@localhost dockerfile]# docker exec -it ad0436d6460c bash
[root@ad0436d6460c /]# ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
19: eth0@if20: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
[root@ad0436d6460c /]#
```

（四）自定义网络 1

docker network create --driver bridge testnet

新建一个名称为 testnet 的网桥

```
[root@localhost ~]# docker network create --driver bridge testnet
a436968ba042fb4fcc59d59a606a0ac3473d875eec1f7456181305ae5cc8f824
[root@localhost ~]#
```

brctl show

```
[root@localhost ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
br-a436968ba042         8000.024242bba307       no
docker0         8000.0242fccd1aa5       no
[root@localhost ~]#
```

docker network inspect testnet

```
[
    {
        "Name": "testnet",
        "Id": "a436968ba042fb4fcc59d59a606a0ac3473d875eec1f7456181305ae5cc8f824",
        "Created": "2020-01-22T17:26:41.120415864+08:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
[root@localhost ~]#
```

docker run -it --network=testnet busybox

```
[root@localhost ~]# docker run -it --network=testnet busybox
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:12:00:02
          inet addr:172.18.0.2  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1312 (1.2 KiB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

docker network create --driver bridge --subnet 192.168.1.0/24 --gateway=192.168.1.1 testnet1

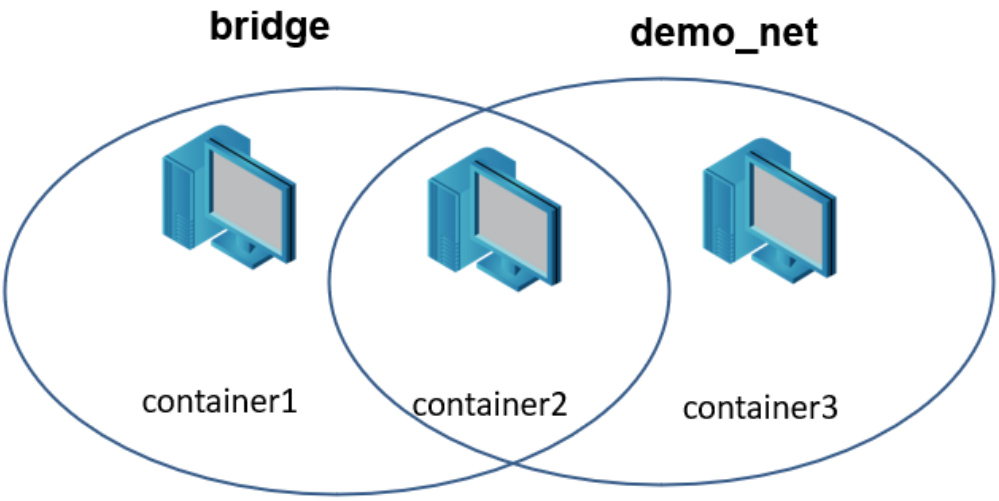
创建自定义网络指定网段和网关

```
[root@localhost ~]# docker network create --driver bridge --subnet 192.168.1.0/24 --gateway=192.168.1.1 testnet1
2999eadaae892f7e585b7ac4ea6ebe487fca4b7cbf94e7288971834fd34ec6ea
[root@localhost ~]#
```

brctl show

```
[root@localhost ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
br-2999eadaae89         8000.02428cabf823       no
br-a436968ba042         8000.024242bba307       no
docker0         8000.0242fccd1aa5       no
[root@localhost ~]#
```

（四）自定义网络 2



docker run -itd --name=container1 busybox

docker run -itd --name=container2 busybox

```
[root@master ~]# docker run -itd --name=container1 busybox
7185ca7e3a9fd35860de59f6b234aacb90af8702b75a95ec2eb2d3247f470b2b
[root@master ~]# docker run -itd --name=container2 busybox
41eb1a265e212b1d458dd79f5f89f92a2647482ddc0f08a923d6363918cd1343
[root@master ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
41eb1a265e21        busybox             "sh"                8 seconds ago       Up 7 seconds                            container2
7185ca7e3a9f        busybox             "sh"                15 seconds ago      Up 13 seconds                           container1
```

docker network create -d bridge --subnet 172.25.0.0/16 demo_net

-d 指使用 Dockerbridge(桥接)方式的网络,如果没有加默认的也是桥接网络

docker network ls

```
[root@master ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
1a066ac145a4        bridge              bridge              local
effda6a86d56        demo_net            bridge              local
5e1cf339e32a        host                host                local
45e670ea6a72        none                null                local
[root@master ~]#
```

docker network connect demo_net container2

docker network inspect demo_net

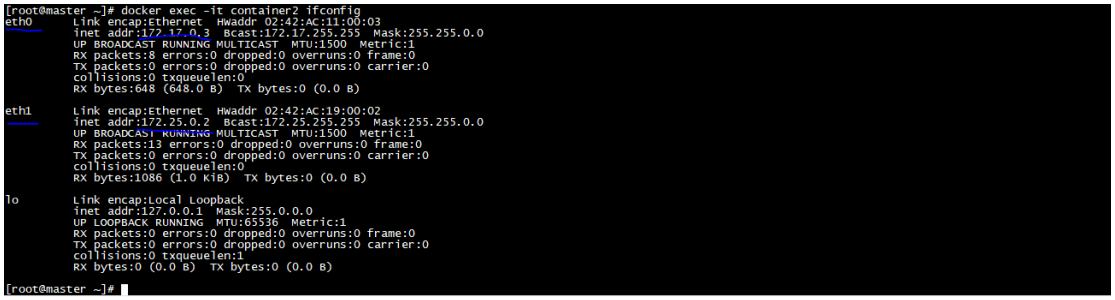
```
[root@master ~]# docker network connect demo_net container2
[root@master ~]# docker network inspect demo_net
[
    {
        "Name": "demo_net",
        "Id": "effda6a86d56b69bbe7731b20506518df5a71284b1a8b6672a41e0358b4ba7a2",
        "Created": "2020-11-01T21:56:30.817856377-05:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.25.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "41eb1a265e212b1d458dd79f5f89f92a2647482ddc0f08a923d6363918cd1343": {
                "Name": "container2",
                "EndpointID": "3b1bc17c7b12ca73e5ca401a766b82c2d9236e7544b6e8dfba6133ef71234b4b",
                "MacAddress": "02:42:ac:19:00:02",
                "IPv4Address": "172.25.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

docker run --network=demo_net --ip=172.25.3.3 -itd --name=container3 busybox

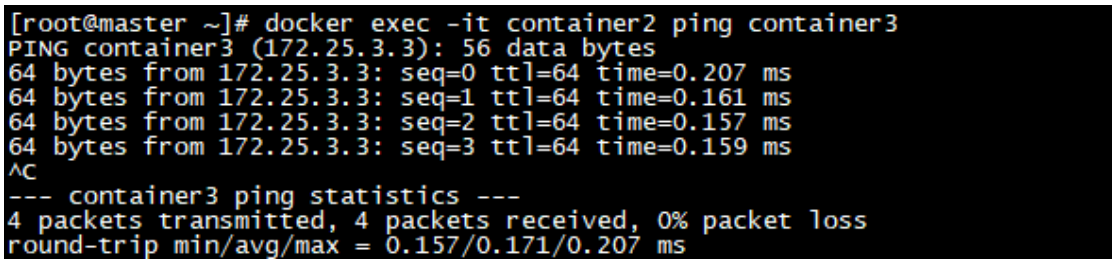
```
[root@master ~]# docker run --network=demo_net --ip=172.25.3.3 -itd --name=container3 busybox
30b08bcac21a17976c3b5e8db2e0d000617f4b0839bed04045e91ae769b10402
[root@master ~]#
```

docker network inspect demo_net

docker exec -it container2 ifconfig



docker exec -it container2 ping container3



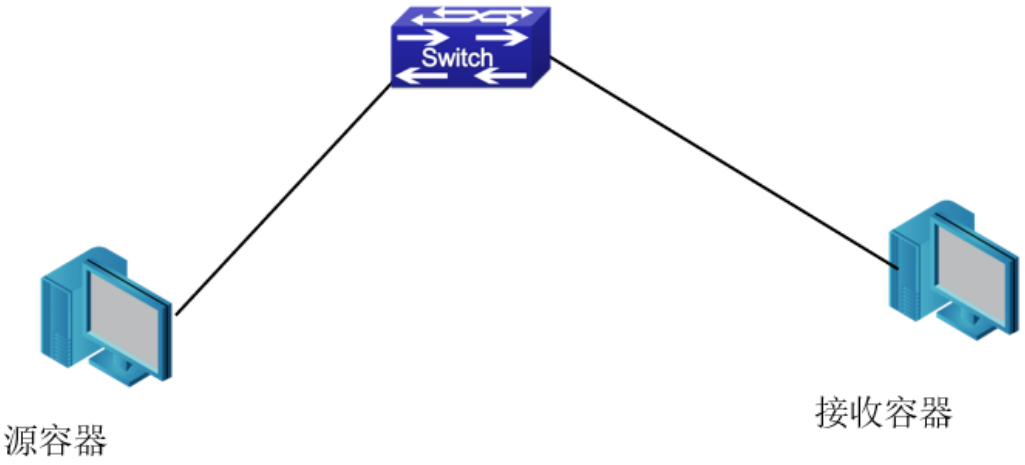
docker network inspect bridge

docker exec -it container2 ping 172.17.0.2

```
[root@master ~]# docker exec -it container2 ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: seq=0 ttl=64 time=0.114 ms
64 bytes from 172.17.0.2: seq=1 ttl=64 time=0.159 ms
64 bytes from 172.17.0.2: seq=2 ttl=64 time=0.157 ms
64 bytes from 172.17.0.2: seq=3 ttl=64 time=0.159 ms
64 bytes from 172.17.0.2: seq=4 ttl=64 time=0.157 ms
64 bytes from 172.17.0.2: seq=5 ttl=64 time=0.141 ms
64 bytes from 172.17.0.2: seq=6 ttl=64 time=0.147 ms
64 bytes from 172.17.0.2: seq=7 ttl=64 time=0.157 ms
64 bytes from 172.17.0.2: seq=8 ttl=64 time=0.156 ms
^C
--- 172.17.0.2 ping statistics ---
9 packets transmitted, 9 packets received, 0% packet loss
round-trip min/avg/max = 0.114/0.149/0.159 ms
```

（五）容器连接永久

Docker 容器运行互相之间就可以联通，但容器重新启动后 ip 地址就改变，在生产环境不利于管理。



1.创建源容器

docker run -itd --name busyboxtest1 busybox /bin/sh

```
[root@centos71611 ~]# docker run -itd --name busyboxtest1 busybox  /bin/sh
101eab2aa99901334c704c59f6c28de269b08d9d083412bd945876380cc1fba4
[root@centos71611 ~]#
```

docker exec -it 101eab2aa999 /bin/sh

进入容器，查看容器 IP 地址，可以看到容器的 IP 地址 172.16.0.2

```
[root@centos71611 ~]# docker exec -it 101eab2aa999 /bin/sh
/ # ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
6: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
/ #
```

2.创建被接收容器

（1）ctrl+p，然后 ctrl+q 退出容器，这样保证容器还在运行

```
[root@localhost ~]# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
3916ad6bda15        centos              "/bin/bash"         About a minute ago  Up About a minute                       keen_raman
[root@localhost ~]#
```

（2）# # docker run -itd --name busyboxtest2 --link busyboxtest1:busybox11111 busybox /bin/sh

busyboxtest1 是上面运行的容器名称，源容器

busybox11111 是 busyboxtest1 别名

可以看到容器的 IP 地址 172.16.0.3

```
[root@centos71611 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
110fe05b8db6        busybox             "/bin/sh"           47 seconds ago      Up 46 seconds                           busyboxtest2
101eab2aa999        busybox             "/bin/sh"           7 minutes ago       Up 7 minutes                            busyboxtest1
[root@centos71611 ~]# docker exec -it 110fe05b8db6  /bin/sh
/ # ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.3/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
/ #
```

在容器 busyboxtest2 操作

[root@82cdc5846fbe /]# cat /etc/hosts

```
/ # cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      busybox11111 101eab2aa999 busyboxtest1
172.17.0.3      110fe05b8db6
/ #
```

```
/ # ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: seq=0 ttl=64 time=0.133 ms
64 bytes from 172.17.0.2: seq=1 ttl=64 time=0.055 ms
^C
--- 172.17.0.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.055/0.094/0.133 ms
/ # ping busybox11111
PING busybox11111 (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: seq=0 ttl=64 time=0.068 ms
64 bytes from 172.17.0.2: seq=1 ttl=64 time=0.071 ms
^C
--- busybox11111 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.068/0.069/0.071 ms
/ # ping 101eab2aa999
PING 101eab2aa999 (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: seq=0 ttl=64 time=0.158 ms
64 bytes from 172.17.0.2: seq=1 ttl=64 time=0.078 ms
^C
--- 101eab2aa999 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.078/0.118/0.158 ms
/ # ping busyboxtest1
PING busyboxtest1 (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: seq=0 ttl=64 time=0.051 ms
64 bytes from 172.17.0.2: seq=1 ttl=64 time=0.153 ms
^C
--- busyboxtest1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.051/0.102/0.153 ms
/ #
```

[root@fb75438e3095 /]# env

```
/ # env
HOSTNAME=110fe05b8db6
SHLVL=1
HOME=/root
TERM=xterm
BUSYBOX11111_NAME=/busyboxtest2/busybox11111
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
PWD=/
/ #
```

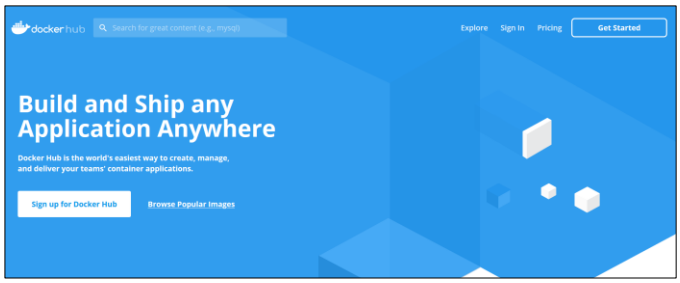
ping www.baidu.com

```
/ # ping www.baidu.com
PING www.baidu.com (182.61.200.6): 56 data bytes
64 bytes from 182.61.200.6: seq=0 ttl=52 time=5.678 ms
64 bytes from 182.61.200.6: seq=1 ttl=52 time=5.602 ms
64 bytes from 182.61.200.6: seq=2 ttl=52 time=13.958 ms
^C
--- www.baidu.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 5.602/8.412/13.958 ms
/ #
```

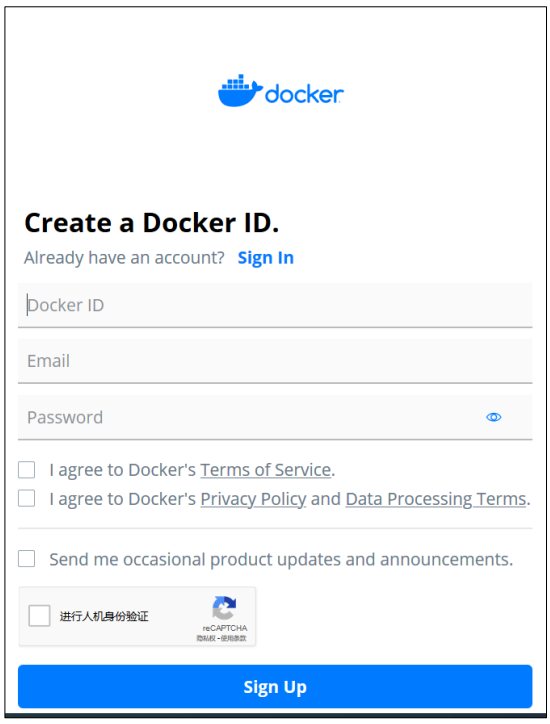
九、Docker 仓库

（一）公有仓库申请

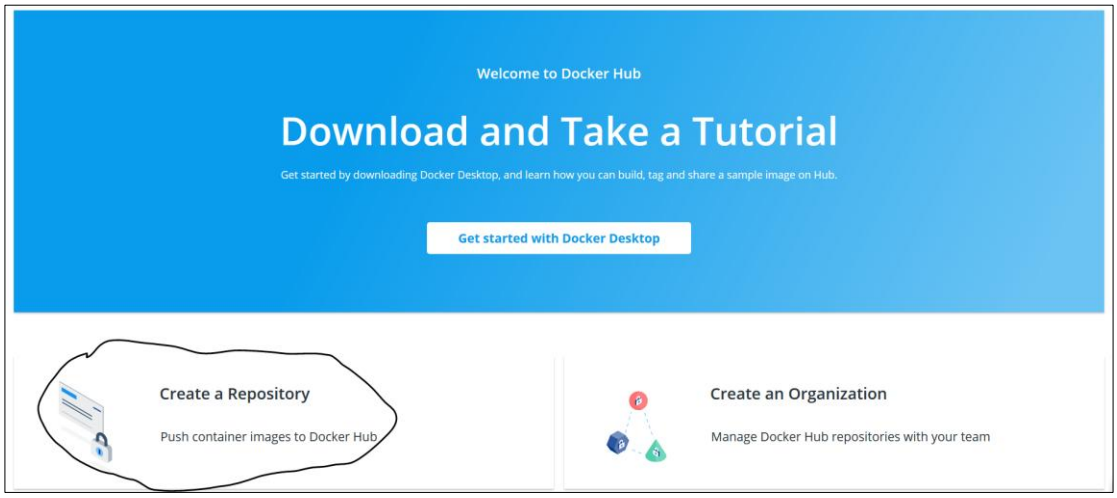
1.到 https://hub.docker.com/注册，既可以建立是公有也可以私有仓库。



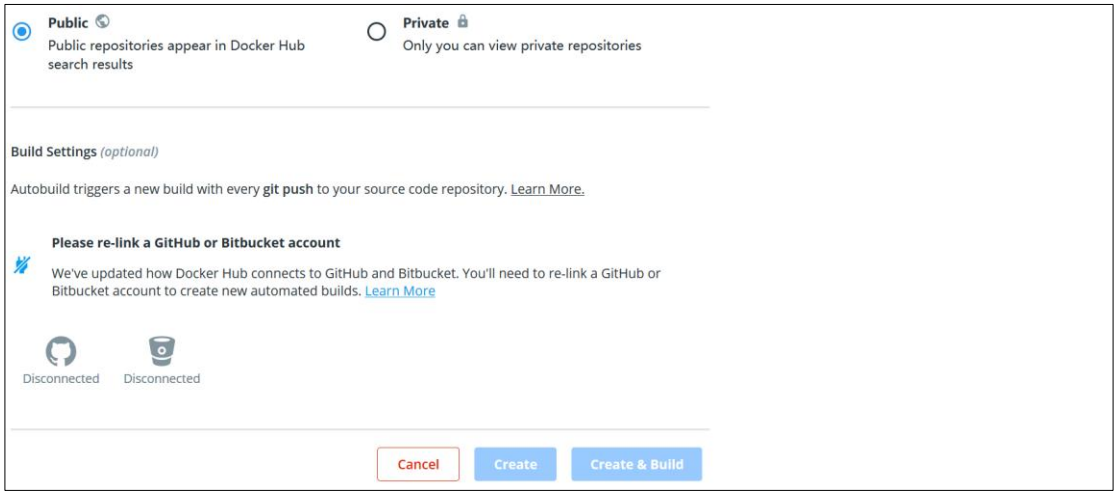
2.注册



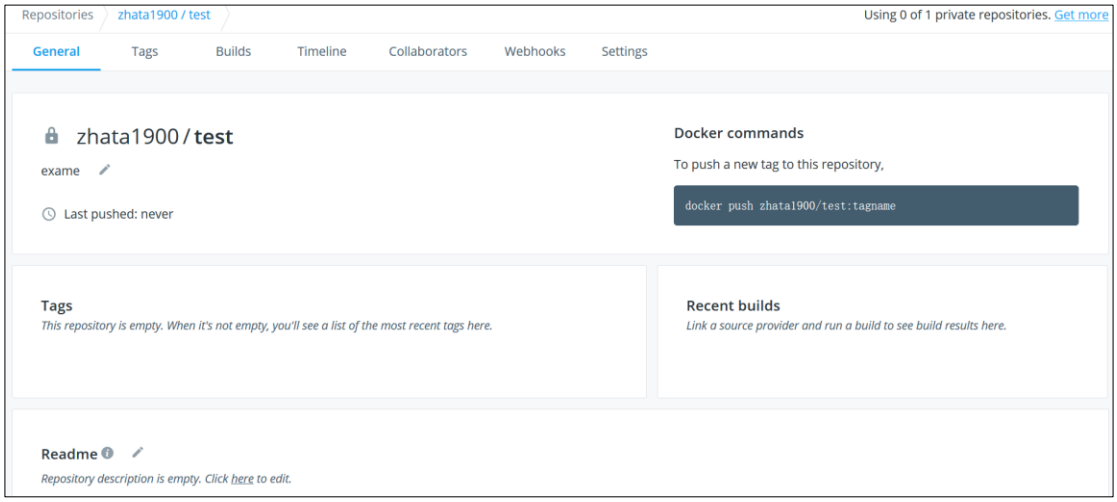
3.建立仓库



4.建立公有或者私有仓库



5.私有仓库建立成功



6.推送自定义镜像到私库

docker login

```
[root@localhost ~]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to crea
te one.
Username: zhata1900
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@localhost ~]#
```

docker images

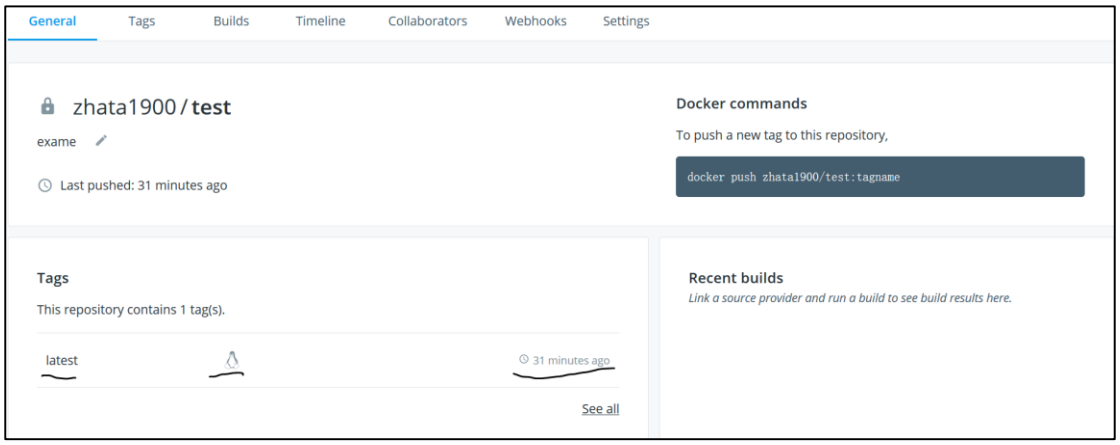
```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
zhata1900/test      latest              35da7584e7d5        22 seconds ago      152MB
ceshi/static_web    latest              3f89136d2a7e        3 hours ago         152MB
<none>              <none>              cb235a472003        4 hours ago         92MB
<none>              <none>              31ea0e7eaade        4 hours ago         152MB
<none>              <none>              07f15817012f        4 hours ago         126MB
centos              latest              470671670cac        3 days ago          237MB
ubuntu              latest              ccc6e87d482b        5 days ago          64.2MB
nginx               latest              c7460dfcab50        11 days ago         126MB
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        3 weeks ago         1.22MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
[root@localhost ~]#
```

docker push zhata1900/test

给自己镜像命名的时候格式必须是: docker 注册用户名/镜像名，否则上传不上去

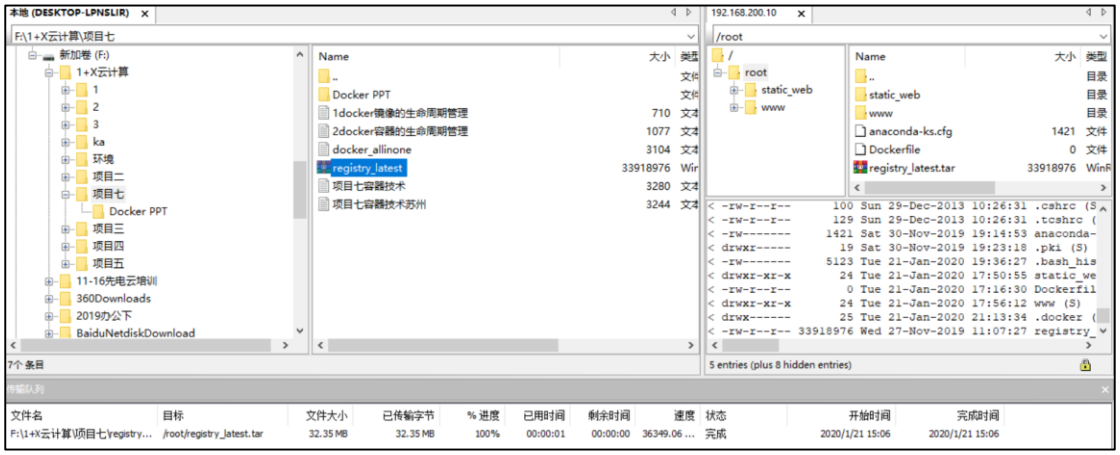
```
[root@localhost ~]# docker push zhata1900/test
The push refers to repository [docker.io/zhata1900/test]
c856e6004c48: Pushed
d612cb1ccfb1: Pushed
c3b6fb04c0c5: Pushed
f55aa0bd26b8: Pushed
1d0dfb259f6a: Pushed
21ec61b65b20: Pushed
43c67172d1d1: Pushed
Put https://registry-1.docker.io/v2/zhata1900/test/blobs/uploads/ae73cad9-9662-413b-a24f-77e0deb23be9?_state=1GtgeXfsFBM8AjHq8Ei1EKoY5QYh9w7usv
kwO24k4Kp7Ik5hBWUTOj16aGFOYTE5MDAvdGVzdCIsIlVvSUQiOiJhZTczY2FkOS05NjYyLTQxM2ItYTI0Zi03N2UwZGViMjNiZTkiLCJPZmZzZXQiOjQxOTksIlN0YXJ0ZWRBdCI6IjIwM
jATMDEtMjFUMDU6MjU6NThaInO%3D&digest=sha256%3A35da7584e7d55ae222d6a45e434b9ccfa4ae7e6e641542f2ecc8a5cecb570a18: dial tcp: lookup registry-1.doc
ker.io on 8.8.8.8:53: read udp 192.168.200.10:60490->8.8.8.8:53: i/o timeout
[root@localhost ~]#
```

验证



（二）本地仓库搭建（本地私有仓库-离线）

1. registry 上传至 linux 的/root



2. 运行 registry

docker load < registry_latest.tar

```
[root@localhost ~]# docker load < registry_latest.tar
011b303988d2: Loading layer [==================================================>]  5.05MB/5.05MB
d57f828d06ea: Loading layer [==================================================>]  1.631MB/1.631MB
a049b9c716b3: Loading layer [==================================================>]  27.21MB/27.21MB
481c807467a1: Loading layer [==================================================>]  3.584kB/3.584kB
9b728062fb6d: Loading layer [==================================================>]  2.048kB/2.048kB
Loaded image: registry:latest
[root@localhost ~]#
```

3. 运行一个仓库服务容器

docker run -d -p 5000:5000 --restart=always --name registry registry

```
[root@localhost ~]# docker run -d  -p 5000:5000 --restart=always --name registry registry
c2bffc73e568f6e34b56ee3ba12a8f69c04b3da622969d26621d32c090e3323f
[root@localhost ~]#
```

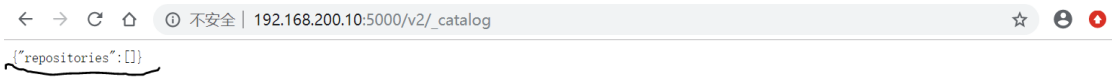
4.查看运行结果

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
c2bffc73e568        registry            "/entrypoint.sh /etc…"   34 seconds ago      Up 34 seconds       0.0.0.0:5000->5000/tcp   registry
4e46c06f237a        nginx               "nginx -g 'daemon of…"   49 minutes ago      Up 49 minutes       0.0.0.0:80->80/tcp       zen_swartz
[root@localhost ~]#
```

5. 访问测试仓库

http://192.168.200.10:5000/v2/_catalog



或者使用 curl 测试

先安装 curl

yum install -y net-tools

测试

```
[root@localhost ~]# curl http://192.168.200.10:5000/v2/_catalog
{"repositories":[]}
[root@localhost ~]#
```

6.查看本地镜像

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
zhata1900/test      latest              35da7584e7d5        2 hours ago         152MB
ceshi/static_web    latest              3f89136d2a7e        5 hours ago         152MB
<none>              <none>              cb235a472003        5 hours ago         92MB
<none>              <none>              31ea0e7eaade        6 hours ago         152MB
<none>              <none>              07f15817012f        6 hours ago         126MB
centos              latest              470671670cac        3 days ago          237MB
ubuntu              latest              ccc6e87d482b        5 days ago          64.2MB
nginx               latest              c7460dfcab50        11 days ago         126MB
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        3 weeks ago         1.22MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
registry            latest              c9bd19d022f6        3 years ago         33.3MB
[root@localhost ~]#
```

7.# vi /etc/docker/daemon.json

加入一行

```
{"registry-mirrors": ["192.168.200.10:5000"]}
~
~
```

8.重新启动 docker 服务

systemctl restart docker

9.给镜像打下如下图格式标记

docker tag hello-world 127.0.0.1:5000/hello-world //必须是：**ip:5000/名称** 这种格式

docker images

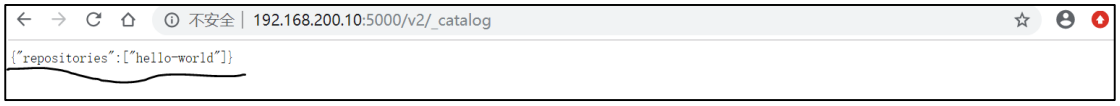
```
[root@localhost ~]# docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
127.0.0.1:5000/hello-world    latest              fce289e99eb9        12 months ago       1.84kB
hello-world                   latest              fce289e99eb9        12 months ago       1.84kB
registry                      latest              c9bd19d022f6        3 years ago         33.3MB
```

10.上传到私有仓库

docker push 127.0.0.1:5000/hello-world

```
[root@localhost ~]#  docker push 127.0.0.1:5000/hello-world
The push refers to repository [127.0.0.1:5000/hello-world]
af0b15c8625b: Pushed
latest: digest: sha256:92c7f9c92844bbbb5d0a101b22f7c2a7949e40f8ea90c8b3bc396879d95e899a size: 524
[root@localhost ~]#
```

11.验证



或者

```
[root@localhost ~]# curl http://192.168.200.10:5000/v2/_catalog
{"repositories":["hello-world"]}
[root@localhost ~]#
```

（三）本地仓库搭建（本地私有仓库-在线）

1. 下载 registry

docker pull registry

```
[root@localhost ~]# docker pull registry
Using default tag: latest
latest: Pulling from library/registry
c87736221ed0: Pulling fs layer
1cc8e0bb44df: Pulling fs layer
54d33bcb37f5: Pulling fs layer
e8afc091c171: Waiting
b4541f6d3db6: Waiting
latest: Pulling from library/registry
c87736221ed0: Pull complete
1cc8e0bb44df: Pull complete
54d33bcb37f5: Pull complete
e8afc091c171: Pull complete
b4541f6d3db6: Pull complete
Digest: sha256:8004747f1e8cd820a148fb7499d71a76d45ff66bac6a29129bfdbfdc0154d146
Status: Downloaded newer image for registry:latest
docker.io/library/registry:latest
[root@localhost ~]#
```

2.运行一个仓库服务容器

docker run -d -p 5000:5000 --restart=always --name registry registry

```
[root@localhost ~]# docker run -d  -p 5000:5000 --restart=always --name registry registry
c2bffc73e568f6e34b56ee3ba12a8f69c04b3da622969d26621d32c090e3323f
[root@localhost ~]#
```

3.查看运行结果

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
c2bffc73e568        registry            "/entrypoint.sh /etc…"   34 seconds ago      Up 34 seconds       0.0.0.0:5000->5000/tcp   registry
4e46c06f237a        nginx               "nginx -g 'daemon of…"   49 minutes ago      Up 49 minutes       0.0.0.0:80->80/tcp       zen_swartz
[root@localhost ~]#
```

4. 访问测试仓库

http://192.168.200.10:5000/v2/_catalog

或者使用 curl 测试

先安装 curl

yum install -y net-tools

测试

```
[root@localhost ~]# curl http://192.168.200.10:5000/v2/_catalog
{"repositories":[]}
[root@localhost ~]#
```

5.查看本地镜像

docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
zhata1900/test      latest              35da7584e7d5        2 hours ago         152MB
ceshi/static_web    latest              3f89136d2a7e        5 hours ago         152MB
<none>              <none>              cb235a472003        5 hours ago         92MB
<none>              <none>              31ea0e7eaade        6 hours ago         152MB
<none>              <none>              07f15817012f        6 hours ago         126MB
centos              latest              470671670cac        3 days ago          237MB
ubuntu              latest              ccc6e87d482b        5 days ago          64.2MB
nginx               latest              c7460dfcab50        11 days ago         126MB
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        3 weeks ago         1.22MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
registry            latest              c9bd19d022f6        3 years ago         33.3MB
[root@localhost ~]#
```

6.# vi /etc/docker/daemon.json

加入一行

```
{"registry-mirrors": ["192.168.200.10:5000"]}
~
~
```

7.重新启动 docker 服务

systemctl restart docker

8.给镜像打下如下图格式标记

docker tag hello-world 127.0.0.1:5000/hello-world //必须是：**ip:5000/名称** 这种格式

docker images

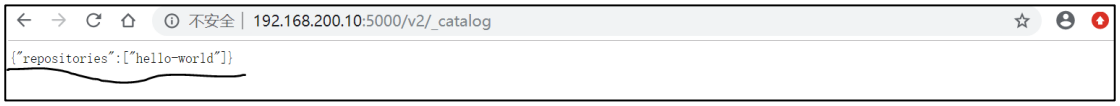
```
[root@localhost ~]# docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
127.0.0.1:5000/hello-world  latest              fce289e99eb9        12 months ago       1.84kB
hello-world                 latest              fce289e99eb9        12 months ago       1.84kB
registry                    latest              c9bd19d022f6        3 years ago         33.3MB
```

9.上传到私有仓库

docker push 127.0.0.1:5000/hello-world

```
[root@localhost ~]# docker push 127.0.0.1:5000/hello-world
The push refers to repository [127.0.0.1:5000/hello-world]
af0b15c8625b: Pushed
latest: digest: sha256:92c7f9c92844bbbb5d0a101b22f7c2a7949e40f8ea90c8b3bc396879d95e899a size: 524
[root@localhost ~]#
```

10.验证



或者

```
[root@localhost ~]# curl http://192.168.200.10:5000/v2/_catalog
{"repositories":["hello-world"]}
[root@localhost ~]#
```

十、Docker 存储

（一）bind mount 存储

1.# docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              470671670cac        5 days ago          237MB
nginx               latest              c7460dfcab50        13 days ago         126MB
httpd               latest              c2aa7e16edd8        3 weeks ago         165MB
busybox             latest              6d5fcfe5ff17        3 weeks ago         1.22MB
hello-world         latest              fce289e99eb9        12 months ago       1.84kB
[root@localhost ~]#
```

2.# pwd

```
[root@localhost ~]# pwd
/root
[root@localhost ~]#
```

3.

[root@localhost ~]# echo "this is host file" >./ceshi/index.html

[root@localhost ~]# cd ceshi/

[root@localhost ceshi]# ls

index.html

```
[root@localhost ~]# echo "this is host file" >./ceshi/index.html
[root@localhost ~]# cd ceshi/
[root@localhost ceshi]# ls
index.html
[root@localhost ceshi]#
```

4. # cd ..

```
[root@localhost ceshi]# cd ..
```

5. # docker run -d -p 80:80 -v ~/ceshi:/usr/local/apache2/htdocs httpd

```
[root@localhost ~]# docker run -d -p 80:80 -v ~/ceshi:/usr/local/apache2/htdocs httpd
6ffa8532b0b0495c104db01a962f649ef55c81e2dd02a357467e9c9be286127b
[root@localhost ~]#
```

6. http://192.168.200.10/

（二）bind managed volume 存储

bind managed volume 是不指定 mount 源，实际位置在 host 的

/var/lib/docker/volumes 下

docker run -d -p 80:80 -v /usr/local/apache2/htdocs httpd

```
[root@localhost ~]# docker run -d -p 80:80 -v /usr/local/apache2/htdocs httpd
b8f85c6691289c78a62cb3fa229113619dd4953a8d0aa6169f697f44f81d0ae4
[root@localhost ~]#
```

docker inspect b8f85c6691289c78a62cb3fa229113619dd4953a8d0aa6169f697f44f81d0ae4

```
        "Mounts": [
            {
                "Type": "volume",
                "Name": "302b0400d0d8cadf0e8bdf6c421707c43f475a6440a21253146ee69aff673422",
                "Source": "/var/lib/docker/volumes/302b0400d0d8cadf0e8bdf6c421707c43f475a6440a21253146ee69aff673422/_data",
                "Destination": "/usr/local/apache2/htdocs",
                "Driver": "local",
                "Mode": "",
                "RW": true,
                "Propagation": ""
            }
        ],
```

cd
/var/lib/docker/volumes/302b0400d0d8cadf0e8bdf6c421707c43f475a6440a21253146ee69aff673422/_data
echo "this is test" >index.html

```
[root@localhost ~]# cd /var/lib/docker/volumes/302b0400d0d8cadf0e8bdf6c421707c43f475a6440a21253146ee69aff673422/_data
[root@localhost _data]# echo "this is test" >index.html
[root@localhost _data]#
```

http://192.168.200.10/

十一、Docker machine

docker-machine 是集成化创建 docker 工具，在多种平台上快速的安装 Docker 环境。Docker Machine 可以直接管理远程的 docker 主机，不论是虚拟机还是云主机。

Docker Machine 是一种可以让您在虚拟主机上安装 Docker 的工具，并可以使用 docker-machine 命令来管理主机。

Docker Machine 也可以集中管理所有的 docker 主机，比如快速的给 100 台服务器安装上 docker。

使用 docker-machine 命令，您可以启动，检查，停止和重新启动托管主机，也可以升级 Docker 客户端和守护程序，以及配置 Docker 客户端与您的主机进行通信。

安装 Docker Machine 之前你需要先安装 Docker。

Docker Mechine 可以在多种平台上安装使用，包括 Linux 、MacOS 以及 windows。

（一）docker machine 安装

Docker machine 安装帮助网址

https://docs.docker.com/machine/install-machine/

1.确定下载路径

base=https://github.com/docker/machine/releases/download/v0.16.0

```
[root@localhost ~]# base=https://github.com/docker/machine/releases/download/v0.16.0
```

2.下载

curl -L $base/docker-machine-$(uname -s)-$(uname -m) >/usr/local/bin/docker-machine

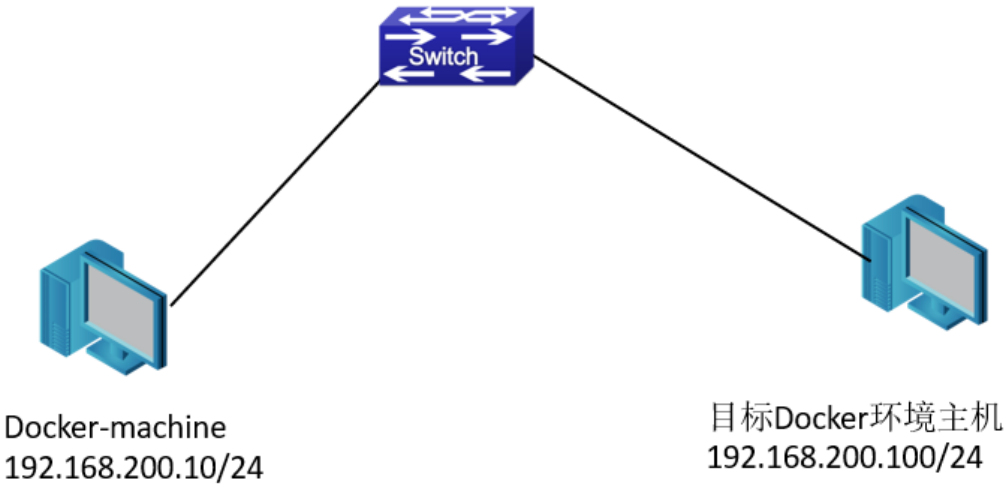
```
[root@localhost ~]# curl -L $base/docker-machine-$(uname -s)-$(uname -m) >/usr/local/bin/docker-machine
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   617    0   617    0     0    574      0 --:--:--  0:00:01 --:--:--   574
100 26.8M  100 26.8M    0     0   739k      0  0:00:37  0:00:37 --:--:--  809k
[root@localhost ~]#
```

3.增加运行属性权限

chmod +x /usr/local/bin/docker-machine

```
[root@localhost ~]#   chmod +x /usr/local/bin/docker-machine
[root@localhost ~]#
```

4.使用 Docker machine 创建远程主机的 docker 环境



（1）在本地计算机生成 ssh 密码

ssh-keygen

默认回车确认

（2）ssh-copy-id **将 key 写到远程机器（192.168200.100）**

ssh-copy-id -i .ssh/id_rsa.pub root@192.168.200.100

root 是远程计算机用户名

```
[root@localhost ~]# ssh-copy-id -i .ssh/id_rsa.pub  root@192.168.200.100
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/id_rsa.pub"
The authenticity of host '192.168.200.100 (192.168.200.100)' can't be established.
ECDSA key fingerprint is SHA256:oyzuYKJUsrE4QH4lNKqkQm8CVki9vHpBYf/Z7wG6t38.
ECDSA key fingerprint is MD5:7b:51:d8:1d:1b:2d:0d:67:e3:c2:bd:81:b5:9f:d6:32.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@192.168.200.100's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@192.168.200.100'"
and check to make sure that only the key(s) you wanted were added.

[root@localhost ~]#
```

（3）创建 Docker machine

docker-machine create --driver generic --generic-ip-address=192.168.200.100 host

（二）查看 docker-machine 运行

docker-machine ls

```
[root@localhost ~]# docker-machine ls
NAME    ACTIVE   DRIVER    STATE     URL                         SWARM   DOCKER    ERRORS
host    -        generic   Running   tcp://192.168.200.100:2376          Unknown   Unable to query docker version: Get https://192.168.200.100:2376/v1.1
5/version: dial tcp 192.168.200.100:2376: connect: no route to host
host1   -        generic   Running   tcp://192.168.200.100:2376          Unknown   Unable to query docker version: Get https://192.168.200.100:2376/v1.1
5/version: dial tcp 192.168.200.100:2376: connect: no route to host
[root@localhost ~]#
```

192.168.200.100 上查看 docker

```
[root@localhost ~]# docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
bdbbaa22dec6: Pull complete
Digest: sha256:6915be4043561d64e0ab0f8f098dc2ac48e077fe23f488ac24b665166898115a
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
busybox             latest              6d5fcfe5ff17        4 weeks ago         1.22MB
[root@localhost ~]# docker run -it busybox
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1016 (1016.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ #
```

十二、搭建私有 git（gogs）

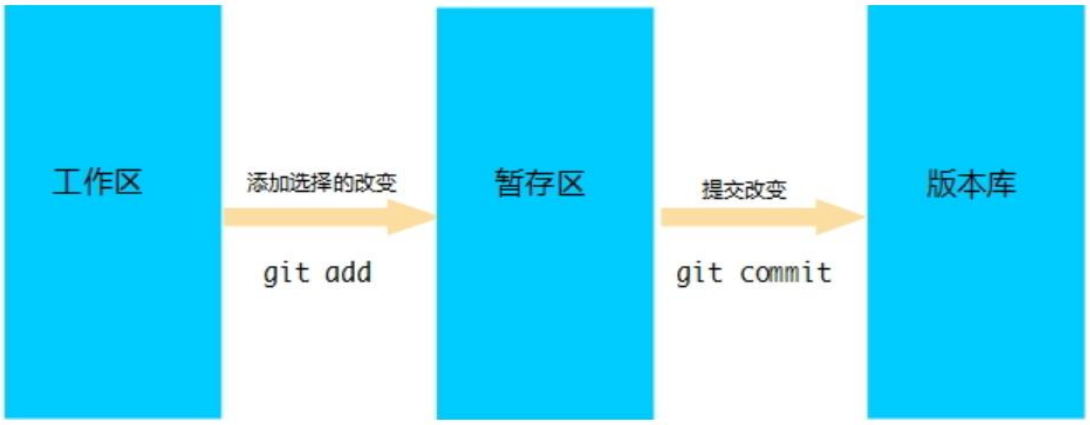
github 是一个基于 git 的代码托管平台，付费用户可以建私人仓库，我们一般的免费用户只能使用公共仓库，也就是代码要公开。Github 由 Chris Wanstrath, PJ Hyett 与 Tom Preston-Werner 三位开发者在 2008 年 4 月创办。迄今拥有 59 名全职员工，主要提供基于 git 的版本托管服务。

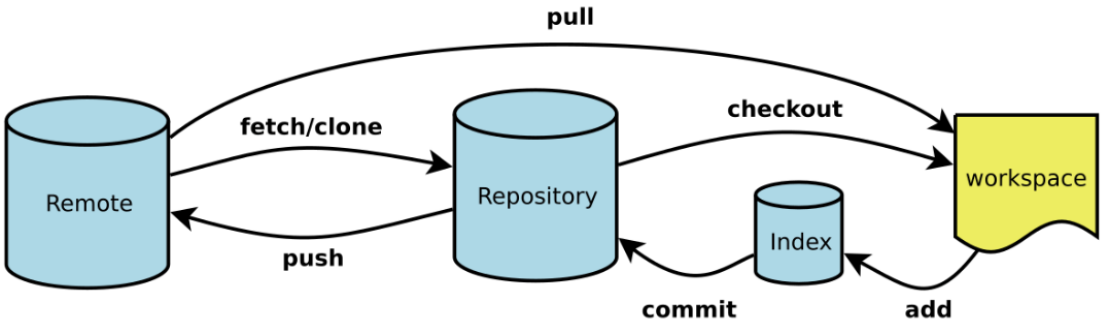
Git 的工作流程一般是：

1.在工作目录中添加、修改文件；

2.将需要进行版本管理的文件放入暂存区域；

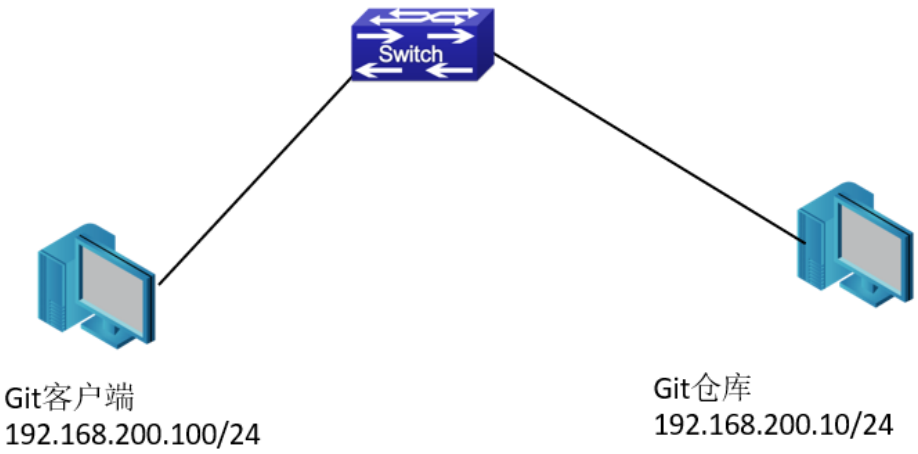
3.将暂存区域的文件提交到 Git 仓库。





Workspace：工作区
Index / Stage：暂存区
Repository：仓库区（或本地仓库）
Remote：远程仓库

（一）私有 git 搭建



1.docker 运行

2.下载 gogs 和 mysql 镜像

docker pull gogs/gogs:latest

```
[root@localhost ~]# docker pull gogs/gogs:latest
latest: Pulling from gogs/gogs
Image docker.io/gogs/gogs:latest uses outdated schema1 manifest format. Please upgrade to a schema2 image for better future compatibility. More information
t https://docs.docker.com/registry/spec/deprecated-schema-v1/
4167d3e14976: Pulling fs layer
704e829dff56: Retrying in 4 seconds
5587acc07e36: Pulling fs layer
601ee68e0a52: Waiting
bf08e966ad0d: Waiting
```

docker pull mysql:latest

3.分别运行 gogs 和 mysql

（1）运行 gogs

docker run -d -p 81:3000 --name gogs gogs/gogs:latest

```
Last login: Fri Feb  7 17:41:31 2020
[root@localhost ~]# docker run -d -p 81:3000 --name gogs gogs/gogs:latest
23e6f34c6f4609013b4e2f5d7d3683186c4ca38f2e98513b01b844783f342b91
[root@localhost ~]#
```

（2）运行 mysql

docker run -d -p 13306:3306 -e MYSQL_ROOT_PASSWORD=000000 --name gogs-mysql mysql:latest

```
[root@localhost ~]# docker run -d -p 13306:3306 -e MYSQL_ROOT_PASSWORD=000000 --name gogs-mysql mysql:latest
5eed7d49aab88e6ec9380f794cc94c4c9509934e5d1131b30a9edd2cb019fe79
[root@localhost ~]#
```

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                NAMES
5eed7d49aab8        mysql:latest        "docker-entrypoint.s…"   59 seconds ago      Up 58 seconds       33060/tcp, 0.0.0.0:13306->3306/tcp   gogs-mysq
23e6f34c6f46        gogs/gogs:latest    "/app/gogs/docker/st…"   3 minutes ago       Up 3 minutes        22/tcp, 0.0.0.0:81->3000/tcp         gogs
[root@localhost ~]#
```

4.建立数据库

（1）进入数据库

docker exec -it gogs-mysql /usr/bin/mysql -uroot -p000000

```
[root@localhost ~]# docker exec -it gogs-mysql /usr/bin/mysql -uroot -p000000
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

（2）建立数据库 gogs

mysql> create database gogs;

```
mysql> create database gogs;
Query OK, 1 row affected (0.00 sec)

mysql>
```

5.安装系统

（1）http://192.168.200.10:81

数据库主机 IP 地址是 docker 宿主机 ip，端口是映射后的 13306

数据库用户密码是 000000

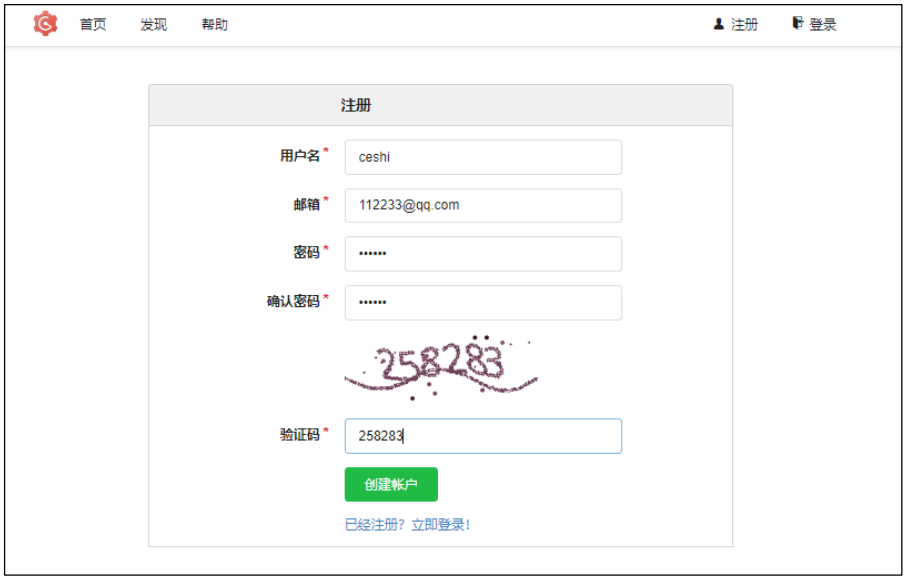
http 应用端口号四映射后 81

应用 URL 是

（2）安装后，再次登录

http://192.168.200.10:81

（3）注册，记住密码



（4）登录

（5）登陆成功

（二）使用私有 git 和 git 命令使用

1.安装 git

yum install -y git

2.查看版本

git –version

```
[root@allinone gittest]# git --version
git version 1.8.3.1
[root@allinone gittest]#
```

3.建立本地文件夹（工作区），进入

mkdir /home/gittest

cd /home/gittest/

```
[root@allinone yum.repos.d]# mkdir /home/gittest
[root@allinone yum.repos.d]# cd /home/gittest/
[root@allinone gittest]# pwd
/home/gittest
[root@allinone gittest]#
```

4.初始化 git

git init

建立了 1 个隐含.git 文件夹，git 相关配置在里面

```
[root@allinone gittest]# git init
Initialized empty Git repository in /home/gittest/.git/
```

5.设置在 git 网站注册的用户名和密码，为将来表示文件用

git config --global user.email 112233@qq.com

git config --global user.name "ceshi"

```
[root@allinone gittest]#  git config --global user.email "112233@qq.com"
[root@allinone gittest]#  git config --global user.name "ceshi"
```

6.设置 git 颜色

git config --global color.ui true

```
[root@allinone gittest]# git config  --global color.ui true
[root@allinone gittest]#
```

7.生成 ssh 密码

（1）# ssh-keygen -t rsa -C “112233@qq.com”

必须是注册时使用的邮箱

格式: ssh-keygen -t rsa －C “youremail@example.com”

```
[root@allinone gittest]# ssh-keygen -t rsa -C "112233@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```

（2）查看公钥内容，复制

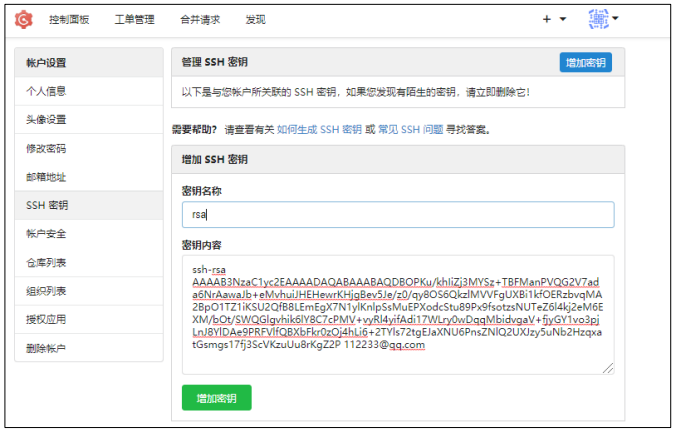
cat /root/.ssh/id_rsa.pub

```
[root@allinone .ssh]# cat  /root/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDBOPKu/khIiZj3MYSz+TBFManPVQG2V7ada6NrAawaJb+eMvhuiJHEHewrKHjgBev5Je/z0/qy8OS6QkzlMVVFgUXBi1kfOEP
MA2BpO1TZ1iKSU2QfB8LEmEgX7N1ylKnlpSsMuEPXodcStu89Px9fSotzsNUTeZ6l4kj2eM6EXM/bOt/SWQGlqvhik6lY8C7cPMV+vyRl4yifAdi17WLryOwDqqMbidvgaV+fjy
o3pjLnJ8YlDAe9PRFVlfQBXbFkrOzOj4hLi6+2TYls72tgEJaXNU6PnsZNlQ2UXJzy5uNb2HzqxatGsmgs17fj3ScVkzuUu8rKgZ2P 112233@qq.com
[root@allinone .ssh]#
```

（3）单击用户设置

（4）单击增加密钥

（5）输入名称和公钥密钥

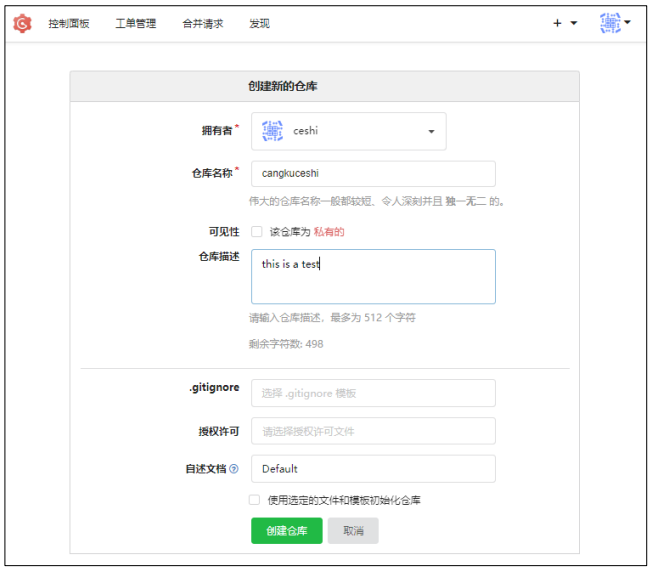


（6）完成 ssh 密钥设置

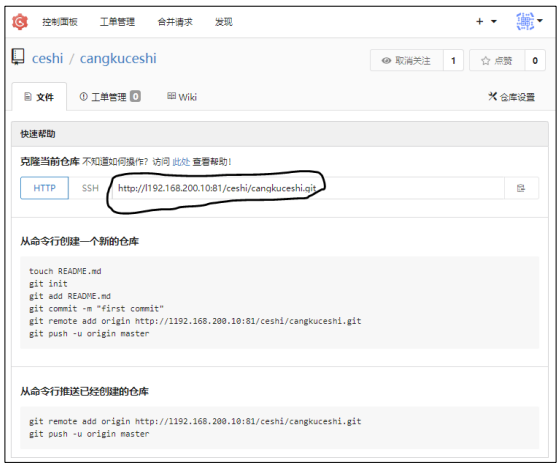
8.建立仓库

（1）控制面板，单击+号

（2）设置新建仓库



（3）新建仓库成功



9.创建一个文件用于测试

touch test.txt

```
[root@allinone gittest]# touch test.txt
[root@allinone gittest]# ls
README.md  READMEtest.md  test.txt
[root@allinone gittest]#
```

10 添加到缓冲区

git add test.txt

```
[root@allinone gittest]# git add test.txt
[root@allinone gittest]#
```

11.将文件提交到 git 仓库（-m 表示添加本次提交的说明，强制要求写的），这只是提交到本地仓库

git commit -m "this is t test"

```
[root@allinone gittest]# git commit -m "this is t test"
[master 8da2027] this is t test
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
[root@allinone gittest]#
```

12. 建立子仓库

git remote add test http://192.168.200.10:81/ceshi/cangkuceshi.git

13.上传到远程仓库

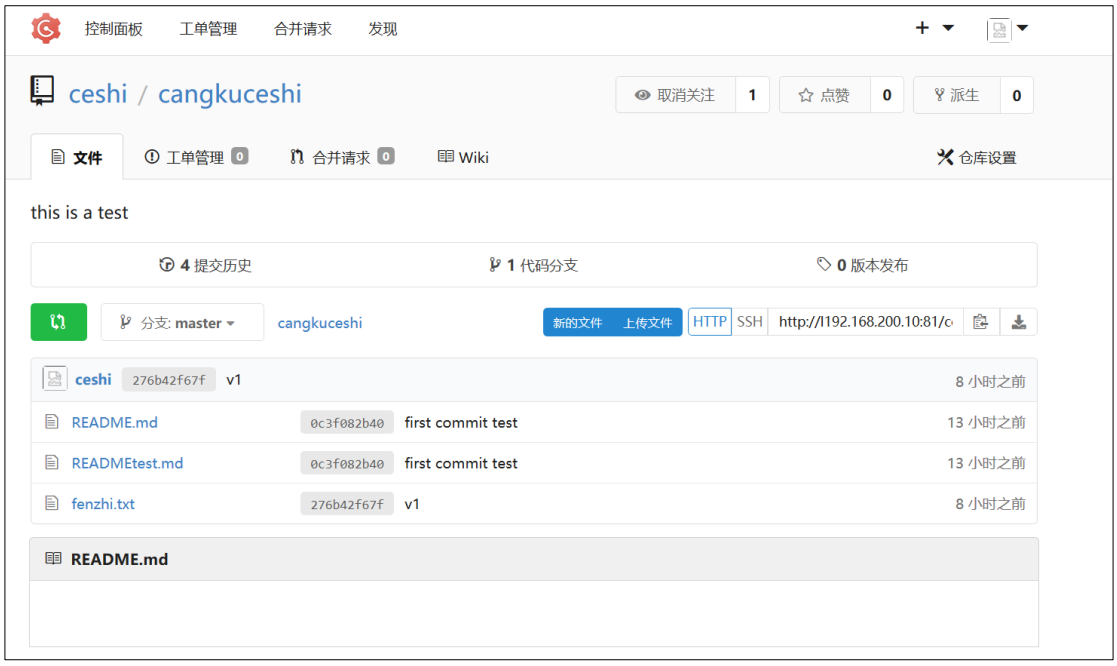
git push test master

git push -u test master

–u 参数，Git 不但会把本地的 master 分支内容推送的远程新的 master 分支，还会把本地的 master 分支和远程的 master 分支关联起来，在以后的推送或者拉取时就可以简化命令。

```
[root@allinone gittest]# git push ceshi master
Counting objects: 7, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 502 bytes | 0 bytes/s, done.
Total 6 (delta 1), reused 0 (delta 0)
Username for 'http://192.168.200.10:81': ceshi
Password for 'http://ceshi@192.168.200.10:81':
To http://192.168.200.10:81/ceshi/cangkuceshi.git
   276b42f..46a892f  master -> master
[root@allinone gittest]#
```

14.远程服务器查看

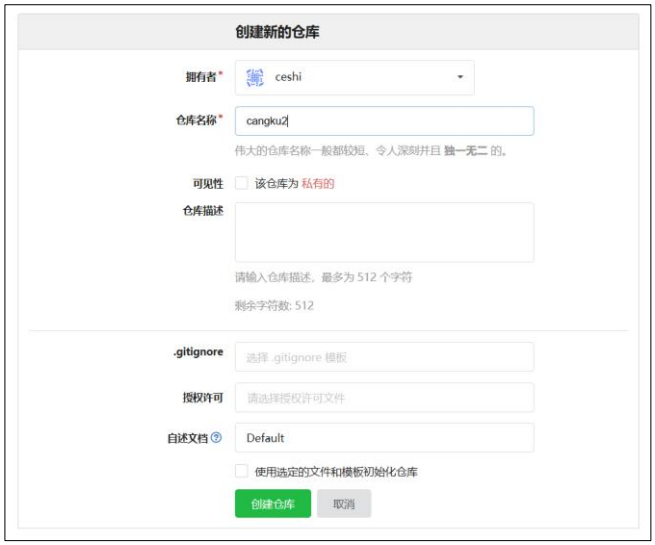


对公网 https://github.com/使用方法同上

15.复制远程服务器仓库到本地

（1）单击创建新的仓库

（2）创建一个新仓库，名字为 cangku2



（3）创建新仓库完成

（4）git clone 克隆到本地库

git clone http://192.168.200.10:81/ceshi/cangku2.git

```
[root@allinone gittest]# git clone http://192.168.200.10:81/ceshi/cangku2.git
Cloning into 'cangku2'...
warning: You appear to have cloned an empty repository.
```

（5）查看本地多了一个文件夹 cangku2

ls

```
[root@allinone gittest]# ls
1.txt  cangku2  ceshi.txt  fenzhi.txt  README.md  READMEtest.md
[root@allinone gittest]#
```

cd cangku2

ls -a

这样就可以使用 cangku2 了

```
[root@allinone cangku2]# ls -a
.  ..  .git
[root@allinone cangku2]#
```

15.其他

（1）查看当前状态

git status　　//通过此命令能够文件的改写状态

```
[root@allinone gittest]# git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   README.md
#
[root@allinone gittest]#
```

（2）查日志

git log

```
[root@allinone gittest]# git log
commit 8da2027c3a141b3b623e51d1f36461c51281e3b3
Author: ceshi <112233@qq.com>
Date:   Thu Feb 6 23:48:08 2020 -0500

    this is t test

commit 0c3f082b40b03d18bb6aa4109accd57ebf7e6003
Author: ceshi <112233@qq.com>
Date:   Thu Feb 6 23:38:00 2020 -0500

    first commit test
[root@allinone gittest]#
```

（3）比较文件修改的不同

git diff

```
[root@allinone gittest]# git diff
diff --git a/test.txt b/test.txt
index 5a4c7a6..15d4c40 100644
--- a/test.txt
+++ b/test.txt
@@ -1 +1,4 @@
 sdwew
+1
+2
+3
[root@allinone gittest]#
```

（4）回退到前一个版本

git reset --hard HEAD^

（5）回退到前两个版本

git reset --hard HEAD^^

（6）回退到前 100 个版本

git reset --hard HEAD~100

（7）查看文件提交版本号

git reflog

```
[root@allinone gittest]# git reflog
8da2027 HEAD@{0}: commit: this is t test
0c3f082 HEAD@{1}: commit (initial): first commit test
[root@allinone gittest]#
```

（8）查看 git 配置

git config --list

```
[root@allinone gittest]# git config --list
user.email=112233@qq.com
user.name=ceshi
color.ui=true
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.testcangku.url=ssh://git@localhost:ceshi/cangkuceshi.git
remote.testcangku.fetch=+refs/heads/*:refs/remotes/testcangku/*
[root@allinone gittest]#
```

（9）删除文件

git rm test.txt

```
[root@allinone gittest]# git rm test.txt
rm 'test.txt'
```

（10）查看远程库

git remote

```
[root@allinone gittest]# git remote
test
testcangku
[root@allinone gittest]#
```

git remote --verbose

```
[root@allinone gittest]# git remote --verbose
test    http://192.168.200.10:81/ceshi/cangkuceshi.git (fetch)
test    http://192.168.200.10:81/ceshi/cangkuceshi.git (push)
testcangku      ssh://git@localhost:ceshi/cangkuceshi.git (fetch)
testcangku      ssh://git@localhost:ceshi/cangkuceshi.git (push)
[root@allinone gittest]#
```

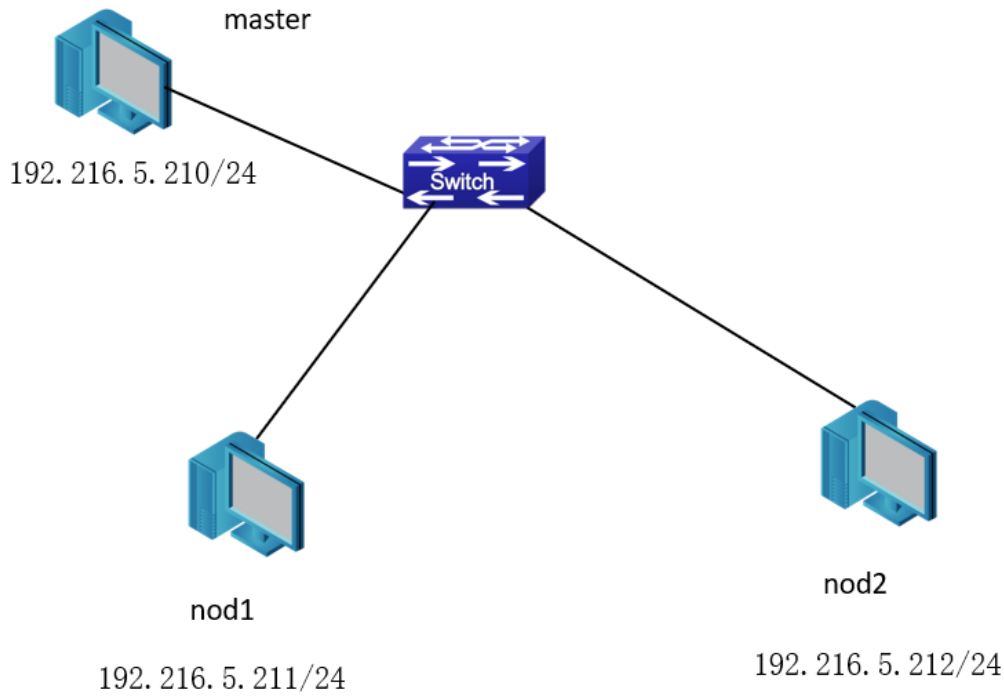
（三）git 学习网址

https://blog.csdn.net/u011535541/article/details/83379151

https://edu.51cto.com/center/course/lesson/index?id=33759

十三、Docker swarm 构建基础

拓扑



Swarm 是 Docker 公司推出的用来管理 docker 集群的平台，几乎全部用 GO 语言来完成的开发的，代码开源在 https://github.com/docker/swarm， 它是将一群 Docker 宿主机变成一个单一的虚拟主机，Swarm 使用标准的 Docker API 接口作为其前端的访问入口，换言之，各种形式的 Docker
Client(compose,docker-py 等)均可以直接与 Swarm 通信，甚至 Docker 本身都可以很容易的与 Swarm 集成，这大大方便了用户将原本基于单节点的系统移植到 Swarm 上，同时 Swarm 内置了对 Docker 网络插件的支持，用户也很容易的部署跨主机的容器集群服务。

Docker Swarm 和 Docker Compose 一样，都是 Docker 官方容器编排项目，但不同的是，Docker Compose 是一个在单个服务器或主机上创建多个容器的工具，而 Docker Swarm 则可以在多个服务器或主机上创建容器集群服务，对于微服务的部署，显然 Docker Swarm 会更加适合。

从 Docker 1.12.0 版本开始，Docker Swarm 已经包含在 Docker 引擎中（docker swarm），并且已经内置了服务发现工具，我们就不需要像之前一样，再配置 Etcd 或者 Consul 来进行服务发现配置了。

Swarm deamon 只是一个调度器(Scheduler)加路由器(router),Swarm 自己不运行容器，它只是接受 Docker 客户端发来的请求，调度适合的节点来运行容器，这就意味着，即使 Swarm 由于某些原因挂掉了，集群中的节点也会照常运行，放 Swarm 重新恢复运行之后，他会收集重建集群信息。

一、主机名配置

1.根据拓扑图分别设置三台主机主机名

#hostname set-hostname master

#hostname set-hostname nod1

#hostname set-hostname nod2

2.分别在三台主机设置主机名解析

vi /etc/hosts

192.216.5.210 master

192.216.5.211 nod1

192.216.5.211 nod2

二、安装 docker

三、关闭防火墙和 selinux

1.关闭防火墙

systemctl stop firewalld 停止防火墙运行

systemctl disable firewalld 禁止防火墙开机运行

2.关闭 selinux

setenforce 0 临时关闭

vi /etc/selinux/config

将 SELINUX=enforcing 改为 SELINUX=disabled（**需要 linux 重新启动才能起作用**）

四、部署 swarm 集群

docker swarm init --advertise-addr 192.216.5.210

在 master 管理节点操作，创建 swarm 集群，并指定集群的管理地址是 192.216.5.210

```
[root@master ~]# docker swarm init --advertise-addr 192.216.5.210
Swarm initialized: current node (kr4hsilmciuc1if0ihv38ef9d) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-63cuucaovqv33too8avjlmkwp0x9pws8jdgfgg8k8g7bk70n1l-3oyax7k7lj4jgupc3nq50qk20 192.216.5.210:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

docker swarm join-token worker

在 master 管理节点获取工作节点加入 swarm 群集口令

```
[root@master ~]# docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-63cuucaovqv33too8avjlmkwp0x9pws8jdgfgg8k8g7bk70n1l-3oyax7k7lj4jgupc3nq50qk20 192.216.5.210:2377
```

docker swarm join-token manager

管理节点加入 swarm 集群

```
[root@master ~]# docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-63cuucaovqv33too8avjlmkwp0x9pws8jdgfgg8k8g7bk70n1l-980w3ddnri3wqr2gqwpmhqwo9 192.216.5.210:2377

[root@master ~]#
```

docker swarm join --token SWMTKN-1-63cuucaovqv33too8avjlmkwp0x9pws8jdgfgg8k8g7bk70n1l-3oyax7k7lj4jgupc3nq50qk20 192.216.5.210:2377

工作节点加入 swarm 群集

```
[root@nod1 ~]# docker swarm join --token SWMTKN-1-63cuucaovqv33too8avjlmkwp0x9pws8jdgfgg8k8g7bk70n1l-3oyax7k7lj4jgupc3nq50qk20 192.216.5.210:2377
This node joined a swarm as a worker.
[root@nod1 ~]#
```

docker info

在管理节点查看 swarm 群集信息

```
[root@master ~]# docker info
Client:
 Debug Mode: false

Server:
 Containers: 38
  Running: 18
  Paused: 0
  Stopped: 20
 Images: 17
 Server Version: 19.03.12
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: active
  NodeID: kr4hsilmciuc1if0ihv38ef9d
  Is Manager: true
  ClusterID: 6b1dgcch3tujrwzzl8pnd5eak
  Managers: 1
  Nodes: 3
  Default Address Pool: 10.0.0.0/8
  SubnetSize: 24
  Data Path Port: 4789
  Orchestration:
   Task History Retention Limit: 5
  Raft:
   Snapshot Interval: 10000
   Number of Old Snapshots to Retain: 0
   Heartbeat Tick: 1
   Election Tick: 10
```

docker node ls

查看 swarm 群集节点

```
[root@master ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
kr4hsilmciuc1if0ihv38ef9d *   master              Ready               Active              Leader              19.03.12
jagsmui45vg3ok0n82gkqtjds     nod1                Ready               Active                                  19.03.12
wqes88p1igfb4ybian1jqv2tz     nod2                Ready               Active                                  19.03.12
[root@master ~]#
```

docker node inspect manager

查看 swarm 管理节点

```
[root@master ~]# docker node inspect manager
[]
Status: Error: No such node: manager, Code: 1
[root@master ~]# docker node inspect master
[
    {
        "ID": "kr4hsilmciuc1if0ihv38ef9d",
        "Version": {
            "Index": 9
        },
        "CreatedAt": "2020-09-16T13:35:39.315288245Z",
        "UpdatedAt": "2020-09-16T13:35:39.418417942Z",
        "Spec": {
            "Labels": {},
            "Role": "manager",
            "Availability": "active"
        },
        "Description": {
            "Hostname": "master",
            "Platform": {
                "Architecture": "x86_64",
                "OS": "linux"
            },
            "Resources": {
                "NanoCPUs": 4000000000,
                "MemoryBytes": 4238352384
            },
            "Engine": {
                "EngineVersion": "19.03.12",
                "Plugins": [
                    {
                        "Type": "Log",
                        "Name": "awslogs"
```

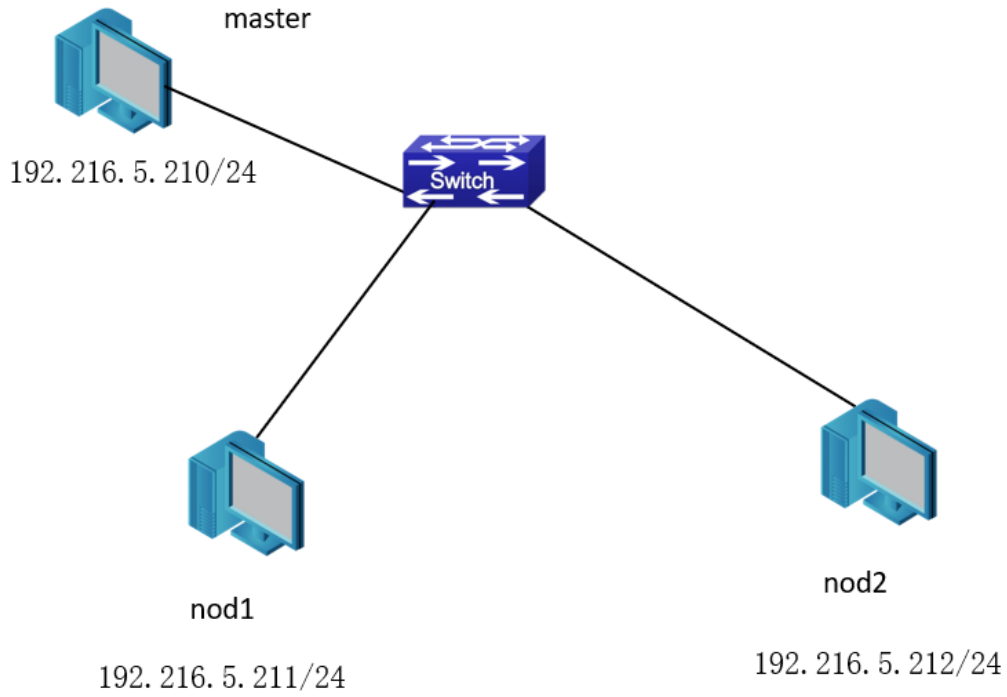
docker node inspect nod1

查看 swarm 工作节点

```
[root@master ~]# docker node inspect nod1
[
    {
        "ID": "jagsmui45vq3ok0n82gkqtjds",
        "Version": {
            "Index": 15
        },
        "CreatedAt": "2020-09-16T13:37:13.292513118Z",
        "UpdatedAt": "2020-09-16T13:37:13.432241253Z",
        "Spec": {
            "Labels": {},
            "Role": "worker",
            "Availability": "active"
        },
        "Description": {
            "Hostname": "nod1",
            "Platform": {
                "Architecture": "x86_64",
                "OS": "linux"
            },
            "Resources": {
                "NanoCPUs": 4000000000,
                "MemoryBytes": 4238368768
            },
            "Engine": {
                "EngineVersion": "19.03.12",
                "Plugins": [
                    {
                        "Type": "Log",
                        "Name": "awslogs"
```

十四、Docker swarm 集群管理

拓扑图



docker node ls

查看 swarm 节点

```
[root@master ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
kr4hsilmciuc1if0ihv38ef9d *   master              Ready               Active              Leader              19.03.12
jagsmui45vq3ok0n82gkqtjds     nod1                Ready               Active                                  19.03.12
wqes88p1igfb4ybian1jqv2tz     nod2                Ready               Active                                  19.03.12
[root@master ~]#
```

变更节点管理状态

docker node update --availability drain master

修改管理节点只具备管理功能

```
[root@master ~]# docker node update --availability drain master
master
[root@master ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
kr4hsilmciuc1if0ihv38ef9d *   master              Ready               Drain               Leader              19.03.12
jagsmui45vq3ok0n82gkqtjds     nod1                Ready               Active                                  19.03.12
wqes88p1igfb4ybianijqv2tz     nod2                Ready               Active                                  19.03.12
[root@master ~]#
```

添加标签元数据

docker node update --label-add GM-IDC-01 nod1

添加标签元数据，为节点 nod1 添加标签 GM-IDC-01，标签就是起到备注作用

docker node inspect nod1

```
[root@master ~]# docker node update --label-add GM-IDC-01 nod1
nod1
[root@master ~]# docker node inspect nod1
[
    {
        "ID": "jagsmui45vq3ok0n82gkqtjds",
        "Version": {
            "Index": 22
        },
        "CreatedAt": "2020-09-16T13:37:13.292513118Z",
        "UpdatedAt": "2020-09-16T14:02:03.856984479Z",
        "Spec": {
            "Labels": {
                "GM-IDC-01": ""
            },
            "Role": "worker",
            "Availability": "active"
        },
        "Description": {
            "Hostname": "nod1",
            "Platform": {
                "Architecture": "x86_64",
                "OS": "linux"
            },
            "Resources": {
                "NanoCPUs": 4000000000,
                "MemoryBytes": 4238368768
            },
            "Engine": {
                "EngineVersion": "19.03.12",
                "Plugins": [
                    {
                        "Type": "Log",
                        "Name": "awslogs"
```

节点提权与降权

docker node promote nod1 nod2

将节点 nod1、nod2 提升为管理节点

```
[root@master ~]# docker node promote nod1 nod2
Node nod1 promoted to a manager in the swarm.
Node nod2 promoted to a manager in the swarm.
[root@master ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
kr4hsilmciuc1if0ihv38ef9d *   master              Ready               Drain               Leader              19.03.12
jagsmui45vq3ok0n82gkqtjds     nod1                Ready               Active              Reachable           19.03.12
wqes88p1igfb4ybianijqv2tz     nod2                Ready               Active              Reachable           19.03.12
[root@master ~]#
```

docker node demote nod1 nod2

将节点 nod1、nod2 降为管理节点

```
[root@master ~]# docker node demote nod1 nod2
Manager nod1 demoted in the swarm.
Manager nod2 demoted in the swarm.
[root@master ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
kr4hsilmciuc1if0ihv38ef9d *   master              Ready               Drain               Leader              19.03.12
jagsmui45vq3ok0n82gkqtjds     nod1                Ready               Active                                  19.03.12
wqes88p1igfb4ybianijqv2tz     nod2                Ready               Active                                  19.03.12
[root@master ~]#
```

退出集群

docker swarm leave

管理节点退出 swarm 集群

```
[root@master ~]# docker swarm leave
Error response from daemon: You are attempting to leave the swarm on a node that is participating as a manager. Removing the last manager erases all current
state of the swarm. Use `--force` to ignore this message.
[root@master ~]#
```

docker swarm leave --force

docker swarm leave

工作节点退出 swarm 集群

```
[root@nod1 ~]#  docker swarm leave
Node left the swarm.
[root@nod1 ~]#
```

docker node ls

```
[root@master ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
kr4hsilmciuc1if0ihv38ef9d *   master              Ready               Drain               Leader              19.03.12
jagsmui45vq3ok0n82gkqtjds     nod1                Down                Active                                  19.03.12
wqes88p1igfb4ybianijqv2tz     nod2                Ready               Active                                  19.03.12
[root@master ~]#
```

docker swarm join --token SWMTKN-1-63cuucaovqv33too8avjlmkwp0x9pws8jdgfgg8k8g7bk70n1l-3oyax7k7lj4jgupc3nq50qk20 192.216.5.210:2377

工作节点重新加入 swarm 群集

```
[root@nod1 ~]# docker swarm join --token SWMTKN-1-63cuucaovqv33too8avjlmkwp0x9pws8jdgfgg8k8g7bk70n1l-3oyax7k7lj4jgupc3nq50qk20 192.216.5.210:2377
This node joined a swarm as a worker.
[root@nod1 ~]#
```

docker node ls

```
[root@master ~]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
kr4hsilmciuc1if0ihv38ef9d *   master              Ready               Drain               Leader              19.03.12
jagsmui45vq3ok0n82gkqtjds     nod1                Down                Active                                  19.03.12
tdl2cqdyi3g82t3v4u3239zft     nod1                Ready               Active                                  19.03.12
wqes88p1igfb4ybian1jqv2tz     nod2                Ready               Active                                  19.03.12
[root@master ~]#
```

创建运行服务

docker service create --replicas 2 --name web nginx

管理节点上操作，创建名称为 web 服务，指定副本数为 2 个

```
[root@master ~]# docker service create --replicas 2 --name web nginx
nfhjgvcx16o1wl4rdvgmvzeh1
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
[root@master ~]#
```

docker service logs -f web

查看运行的服务

```
[root@master ~]# docker service logs -f web
web.1.shytpakswyzz@nod2     | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
web.1.shytpakswyzz@nod2     | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
web.1.shytpakswyzz@nod2     | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
web.1.shytpakswyzz@nod2     | 10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
web.1.shytpakswyzz@nod2     | 10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
web.1.shytpakswyzz@nod2     | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
web.1.shytpakswyzz@nod2     | /docker-entrypoint.sh: Configuration complete; ready for start up
web.2.n6bxqe6bphka@nod1     | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
web.2.n6bxqe6bphka@nod1     | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
web.2.n6bxqe6bphka@nod1     | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
web.2.n6bxqe6bphka@nod1     | 10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/default.conf
web.2.n6bxqe6bphka@nod1     | 10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
web.2.n6bxqe6bphka@nod1     | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
web.2.n6bxqe6bphka@nod1     | /docker-entrypoint.sh: Configuration complete; ready for start up
```

docker service ls

查看已启动所有服务

```
[root@master ~]# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
nfhjgvcx16o1        web                 replicated          2/2                 nginx:latest
[root@master ~]#
```

docker service ps web

查看启动的指定服务

```
[root@master ~]# docker service ps  web
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE            ERROR               PORTS
shytpakswyzz        web.1               nginx:latest        nod2                Running             Running 4 minutes ago
n6bxqe6bphka        web.2               nginx:latest        nod1                Running             Running 4 minutes ago
[root@master ~]#
```

docker ps

查看运行的进程

```
[root@master ~]# docker ps
CONTAINER ID        IMAGE                                                 COMMAND                  CREATED             STATUS              PORTS
NAMES
8cd93660c1a6        fbbc4a1a0e98                                          "/usr/bin/kube-contr…"   48 minutes ago      Up 48 minutes
k8s_calico-kube-controllers_calico-kube-controllers-5447dc9cbf-chlqn_kube-system_b6df959c-ffd2-4356-a645-aece69515e4f_17
6a9659cb8c1d        registry.aliyuncs.com/google_containers/pause:3.2     "/pause"                 48 minutes ago      Up 48 minutes
k8s_POD_calico-kube-controllers-5447dc9cbf-chlqn_kube-system_b6df959c-ffd2-4356-a645-aece69515e4f_41
26f55fddc99a        67da37a9a360                                          "/coredns -conf /etc…"   48 minutes ago      Up 48 minutes
k8s_coredns_coredns-7ff77c879f-q5d6m_kube-system_23809980-0f12-4ae5-a2dc-c3eaa0136162_15
11b17417f5a4        registry.aliyuncs.com/google_containers/pause:3.2     "/pause"                 48 minutes ago      Up 48 minutes
k8s_POD_coredns-7ff77c879f-q5d6m_kube-system_23809980-0f12-4ae5-a2dc-c3eaa0136162_40
50a2bdaa01fe        cc7508d4d2d4                                          "start_runit"            48 minutes ago      Up 48 minutes
k8s_calico-node_calico-node-2qj6z_kube-system_2dcae5d9-de59-4c85-b8e1-8dc58e4cb32a_12
8b1081417 3fd       67da37a9a360                                          "/coredns -conf /etc…"   48 minutes ago      Up 48 minutes
k8s_coredns_coredns-7ff77c879f-6986p_kube-system_280ba7b1-5080-4b49-9930-835de94e3dc5_15
aff2d0e8edf5        registry.aliyuncs.com/google_containers/pause:3.2     "/pause"                 48 minutes ago      Up 48 minutes
k8s_POD_coredns-7ff77c879f-6986p_kube-system_280ba7b1-5080-4b49-9930-835de94e3dc5_33
e24ab38f2d17        43940c34f24f                                          "/usr/local/bin/kube…"   48 minutes ago      Up 48 minutes
k8s_kube-proxy_kube-proxy-g2trr_kube-system_0441501c-c282-4079-a619-3a978e1904c0_12
6cb4b790679c        registry.aliyuncs.com/google_containers/pause:3.2     "/pause"                 48 minutes ago      Up 48 minutes
k8s_POD_calico-node-2qj6z_kube-system_2dcae5d9-de59-4c85-b8e1-8dc58e4cb32a_12
2722e7e299bf        registry.aliyuncs.com/google_containers/pause:3.2     "/pause"                 48 minutes ago      Up 48 minutes
k8s_POD_kube-proxy-g2trr_kube-system_0441501c-c282-4079-a619-3a978e1904c0_12
c35de951551a        74060cea7f70                                          "kube-apiserver --ad…"   49 minutes ago      Up 48 minutes
k8s_kube-apiserver_kube-apiserver-master_kube-system_90438a767b8398ccc7b2f191224afc4f_31
eafaf2c1638e        303ce5db0e90                                          "etcd --advertise-cl…"   49 minutes ago      Up 49 minutes
k8s_etcd_etcd-master_kube-system_f32df599a00876cd2952f02604b8ec62_23
8cc0380a1640        registry.aliyuncs.com/google_containers/pause:3.2     "/pause"                 49 minutes ago      Up 49 minutes
k8s_POD_etcd-master_kube-system_f32df599a00876cd2952f02604b8ec62_13
f376019f9ea5        d3e55153f52f                                          "kube-controller-man…"   49 minutes ago      Up 49 minutes
k8s_kube-controller-manager_kube-controller-manager-master_kube-system_07c76f1314a6ff7850223d35319ebe9a_14
c72f68cf4250        a31f78c7c8ce                                          "kube-scheduler --au…"   49 minutes ago      Up 49 minutes
k8s_kube-scheduler_kube-scheduler-master_kube-system_35d86d4a27d4ed2186b3ab641e946a02_14
70892e47c29c        registry.aliyuncs.com/google_containers/pause:3.2     "/pause"                 49 minutes ago      Up 49 minutes
k8s_POD_kube-apiserver-master_kube-system_90438a767b8398ccc7b2f191224afc4f_13
da2ce5eb4e2b        registry.aliyuncs.com/google_containers/pause:3.2     "/pause"                 49 minutes ago      Up 49 minutes
k8s_POD_kube-scheduler-master_kube-system_35d86d4a27d4ed2186b3ab641e946a02_12
fdac128fcfaf        registry.aliyuncs.com/google_containers/pause:3.2     "/pause"                 49 minutes ago      Up 49 minutes
k8s_POD_kube-controller-manager-master_kube-system_07c76f1314a6ff7850223d35319ebe9a_13
[root@master ~]#
```

docker service inspect web

查看服务详细信息

```
[root@master ~]# docker service inspect web
[
    {
        "ID": "nfhjgvcx16o1wl4rdvgmvzeh1",
        "Version": {
            "Index": 56
        },
        "CreatedAt": "2020-09-16T14:17:37.969814396Z",
        "UpdatedAt": "2020-09-16T14:17:37.969814396Z",
        "Spec": {
            "Name": "web",
            "Labels": {},
            "TaskTemplate": {
                "ContainerSpec": {
                    "Image": "nginx:latest@sha256:c628b67d21744fce822d22fdcc0389f6bd763daac23a6b77147d0712ea7102d0",
                    "Init": false,
                    "StopGracePeriod": 10000000000,
                    "DNSConfig": {},
                    "Isolation": "default"
                },
                "Resources": {
                    "Limits": {},
                    "Reservations": {}
                },
                "RestartPolicy": {
                    "Condition": "any",
                    "Delay": 5000000000,
                    "MaxAttempts": 0
                },
                "Placement": {
                    "Platforms": [
                        {
```

docker service inspect --pretty web

以易于阅读方式显示服务信息

```
[root@master ~]# docker service inspect --pretty web

ID:             nfhjgvcx16o1wl4rdvgmvzeh1
Name:           web
Service Mode:   Replicated
 Replicas:      2
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         nginx:latest@sha256:c628b67d21744fce822d22fdcc0389f6bd763daac23a6b77147d0712ea7102d0
 Init:          false
Resources:
Endpoint Mode:  vip

[root@master ~]#
```

服务扩容与缩容

docker service scale web=3

将运行服务扩容为 3 个

```
[root@master ~]# docker service scale  web=3
web scaled to 3
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
[root@master ~]#
```

docker service ps web

查看服务在哪些节点运行

```
[root@master ~]# docker service ps  web
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE            ERROR               PORTS
shytpakswyzz        web.1               nginx:latest        nod2                Running             Running 11 minutes ago
n6bxqe6bphka        web.2               nginx:latest        nod1                Running             Running 11 minutes ago
cdm5i3kzp3cl        web.3               nginx:latest        nod2                Running             Running 45 seconds ago
[root@master ~]#
```

docker service scale web=1

将运行服务缩容为 1 个

```
[root@master ~]# docker service scale  web=1
web scaled to 1
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[root@master ~]# docker service ps  web
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE            ERROR               PORTS
shytpakswyzz        web.1               nginx:latest        nod2                Running             Running 12 minutes ago
[root@master ~]#
```

docker service rm web

删除服务

```
[root@master ~]# docker service rm  web
web
[root@master ~]# docker service ps  web
no such service: web
[root@master ~]#
```

镜像滚动预更新

docker service create --replicas 2 --name newredis --update-delay 10s redis:3.0.6

新建服务以指定镜像版本运行两个服务，服务间延迟 10s 启动

```
[root@master ~]# docker service create --replicas 2 --name newredis --update-delay 10s redis:3.0.6
osfchpf3v8xb9s8zg8y7blspa
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
[root@master ~]# docker service ps  newredis
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE                ERROR               PORTS
5ntb7uakmiil        newredis.1          redis:3.0.6         nod1                Running             Running about a minute ago
ci4imv9yn7uf        newredis.2          redis:3.0.6         nod2                Running             Running about a minute ago
[root@master ~]#
```

docker service update --image redis:3.0.7 newredis

将服务镜像更新为 3.0.7

```
[root@master ~]# docker service update --image redis:3.0.7 newredis
newredis
overall progress: 2 out of 2 tasks
1/2: running   [==================================================>]
2/2: running   [==================================================>]
verify: Service converged
[root@master ~]# docker service ps  newredis
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE                ERROR               PORTS
5m4qb49rthrj        newredis.1          redis:3.0.7         nod1                Running             Running about a minute ago
5ntb7uakmiil         \_ newredis.1      redis:3.0.6         nod1                Shutdown            Shutdown about a minute ago
qp7cgprhh9i8        newredis.2          redis:3.0.7         nod2                Running             Running 46 seconds ago
ci4imv9yn7uf         \_ newredis.2      redis:3.0.6         nod2                Shutdown            Shutdown about a minute ago
[root@master ~]#
```

数据卷的创建与应用

volume 类型数据卷挂载

docker volume create shujujuan

创建数据卷

```
[root@master ~]# docker volume create shujujuan
shujujuan
[root@master ~]#
```

docker volume ls

查看创建的数据卷

```
[root@master ~]# docker volume ls
DRIVER              VOLUME NAME
local               shujujuan
[root@master ~]#
```

docker service create --mount 'type=volume,src=shujujuan,dst=/usr/share/nginx/html' --replicas 1 --name shujujuan-nginx nginx

绑定数据卷

```
[root@master ~]# docker service create --mount 'type=volume,src=shujujuan,dst=/usr/share/nginx/html' --replicas 1 --name shujujuan-nginx nginx
1ilyc4qndu8m5i2n0skhv80tn
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[root@master ~]#
```

docker service ps shujujuan-nginx

查看服务在哪些节点运行

```
[root@master ~]# docker service  ps shujujuan-nginx
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE           ERROR               PORTS
9s7y7ylq5cpz        shujujuan-nginx.1   nginx:latest        nod1                Running             Running 5 minutes ago
[root@master ~]#
```

docker volume inspect shujujuan

查看数据卷

```
[root@master ~]# docker volume  inspect shujujuan
[
    {
        "CreatedAt": "2020-09-17T07:21:12-04:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/shujujuan/_data",
        "Name": "shujujuan",
        "Options": {},
        "Scope": "local"
    }
]
[root@master ~]#
```

验证数据卷

cd /var/lib/docker/volumes/shujujuan/_data

```
[root@nod1 ~]# cd /var/lib/docker/volumes/shujujuan/_data
[root@nod1 _data]# ls
50x.html  index.html
[root@nod1 _data]#
```

mkdir test01 test02

cd ~

```
[root@nod1 ~]# cd /var/lib/docker/volumes/shujujuan/_data
[root@nod1 _data]# ls
50x.html  index.html
[root@nod1 _data]# mkdir ceshi1 ceshi2
[root@nod1 _data]#
```

docker ps

```
[root@nod1 _data]# docker ps
CONTAINER ID        IMAGE                                                COMMAND                  CREATED             STATUS              PORTS
NAMES
5b19b8732635        nginx:latest                                         "/docker-entrypoint.…"   11 minutes ago      Up 11 minutes       80/tcp
shujujuan-nginx.1.9s7y7ylq5cpzrqy1d4jgbmk4o
fa0129df7777        lizhenliang/java-demo                                "catalina.sh run"        30 minutes ago      Up 30 minutes
k8s_java-demo_test-update-59d8856fdf-mntpq_default_8b5bdf0d-b882-4479-84df-979668cfb0e2_3
6475bc9fd8fa        lizhenliang/java-demo                                "catalina.sh run"        31 minutes ago      Up 31 minutes
k8s_java-demo_test-update-59d8856fdf-5rkw2_default_1bdcaaf5-86f3-4e0a-98b0-ea4986fae40f_1
809af3ab739b        registry.aliyuncs.com/google_containers/pause:3.2   "/pause"                 31 minutes ago      Up 31 minutes
k8s_POD_test-update-59d8856fdf-mntpq_default_8b5bdf0d-b882-4479-84df-979668cfb0e2_3
8f2366c2c0eb        registry.aliyuncs.com/google_containers/pause:3.2   "/pause"                 31 minutes ago      Up 31 minutes
k8s_POD_test-update-59d8856fdf-5rkw2_default_1bdcaaf5-86f3-4e0a-98b0-ea4986fae40f_1
793e476029ba        cc7508d4d2d4                                         "start_runit"            31 minutes ago      Up 31 minutes
k8s_calico-node_calico-node-dxw95_kube-system_93a7efd5-5a3e-42d7-a20f-fa1e17af02dc_8
63d6823f3160        43940c34f24f                                         "/usr/local/bin/kube…"   31 minutes ago      Up 31 minutes
k8s_kube-proxy_kube-proxy-mjpg4_kube-system_73351491-1cbb-4468-8900-4e82a2dad878_8
1ea08b9cee35        registry.aliyuncs.com/google_containers/pause:3.2   "/pause"                 31 minutes ago      Up 31 minutes
k8s_POD_calico-node-dxw95_kube-system_93a7efd5-5a3e-42d7-a20f-fa1e17af02dc_8
4cf5e3819f98        registry.aliyuncs.com/google_containers/pause:3.2   "/pause"                 31 minutes ago      Up 31 minutes
k8s_POD_kube-proxy-mjpg4_kube-system_73351491-1cbb-4468-8900-4e82a2dad878_8
8ebbdd89e8dc        redis:3.0.7                                          "docker-entrypoint.s…"   32 minutes ago      Up 32 minutes       6379/tcp
newredis.1.bdcxw60rg9w56e1xv7ji8ma0w
4a36c4fc3cb4        nginx:latest                                         "/docker-entrypoint.…"   32 minutes ago      Up 32 minutes       80/tcp
newnginx.2.7a131cytvudiesjkxtd98u6y5
[root@nod1 _data]#
```

[root@nod1 _data]# docker exec -it 5b19b8732635 bash

root@5b19b8732635:/# ls /usr/share/nginx/html

```
[root@nod1 _data]# docker exec -it 5b19b8732635  bash
root@5b19b8732635:/# ls /usr/share/nginx/html
50x.html  ceshi1  ceshi2  index.html
root@5b19b8732635:/#
```

mkdir -p /var/vhost/www/test

```
[root@master ~]# mkdir -p /var/vhost/www/test
[root@master ~]#
```

```
[root@nod1 ~]# # mkdir -p /var/vhost/www/test
[root@nod1 ~]#
```

```
[root@nod2 ~]# # mkdir -p /var/vhost/www/test
[root@nod2 ~]#
```

Bind 类型数据卷挂载

docker service create --mount 'type=bind,src=/var/vhost/www/test,dst=/usr/share/nginx/html' --replicas 1 --name nginx-bind nginx

```
[root@master ~]# docker service create --mount 'type=bind,src=/var/vhost/www/test,dst=/usr/share/nginx/html' --replicas 1 --name nginx-bind nginx
nq6nf5v5pe4tgoqappfrmgoxz
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
[root@master ~]#
```

docker service ps nginx-bind

```
[root@master ~]# docker service  ps nginx-bind
ID              NAME              IMAGE          NODE    DESIRED STATE   CURRENT STATE              ERROR
     PORTS
9uujc7wu2lzn    nginx-bind.1      nginx:latest   nod1    Running         Running about a minute ago
yyxxefhxkltz     \_ nginx-bind.1  nginx:latest   nod2    Shutdown        Rejected about a minute ago  "invalid mount config for
ype…"
jjg9ivefv0t7     \_ nginx-bind.1  nginx:latest   nod1    Shutdown        Rejected about a minute ago  "invalid mount config for
ype…"
o16ymgafa2gn     \_ nginx-bind.1  nginx:latest   nod2    Shutdown        Rejected 2 minutes ago       "invalid mount config for
ype…"
zjjm3h5mj6jz     \_ nginx-bind.1  nginx:latest   nod1    Shutdown        Rejected 2 minutes ago       "invalid mount config for
ype…"
[root@master ~]#
```

touch /var/vhost/www/test/ceshi.txt

```
[root@nod1 ~]# touch /var/vhost/www/test/ceshi.txt
[root@nod1 ~]#
```

docker ps -a

```
[root@nod1 ~]# docker ps -a
CONTAINER ID   IMAGE                                              COMMAND                  CREATED          STATUS          PORTS
      NAMES
74b149443cc8   nginx:latest                                       "/docker-entrypoint.…"   7 minutes ago    Up 7 minutes    80/tcp
      nginx-bind.1.9uujc7wu2lznls8brcqqt5dni
48fc86477d56   nginx:latest                                       "/docker-entrypoint.…"   7 minutes ago    Up 7 minutes    80/tcp
      shujujuan-nginx-bind.1.ur8anix6crw7727u1iuj76uhz
5b19b8732635   nginx:latest                                       "/docker-entrypoint.…"   35 minutes ago   Up 35 minutes   80/tcp
      shujujuan-nginx.1.9s7y7ylq5cpzrqy1d4jgbmk4o
fa0129df7777   lizhenliang/java-demo                              "catalina.sh run"        54 minutes ago   Up 54 minutes
      k8s_java-demo_test-update-59d8856fdf-mntpq_default_8b5bdf0d-b882-4479-84df-979668cfb0e2_3
6475bc9fd8fa   lizhenliang/java-demo                              "catalina.sh run"        54 minutes ago   Up 54 minutes
      k8s_java-demo_test-update-59d8856fdf-5rkw2_default_1bdcaaf5-86f3-4e0a-98b0-ea4986fae40f_1
809af3ab739b   registry.aliyuncs.com/google_containers/pause:3.2  "/pause"                 55 minutes ago   Up 55 minutes
      k8s_POD_test-update-59d8856fdf-mntpq_default_8b5bdf0d-b882-4479-84df-979668cfb0e2_3
8f2366c2c0eb   registry.aliyuncs.com/google_containers/pause:3.2  "/pause"                 55 minutes ago   Up 55 minutes
      k8s_POD_test-update-59d8856fdf-5rkw2_default_1bdcaaf5-86f3-4e0a-98b0-ea4986fae40f_1
793e476029ba   cc7508d4d2d4                                       "start_runit"            55 minutes ago   Up 55 minutes
      k8s_calico-node_calico-node-dxw95_kube-system_93a7efd5-5a3e-42d7-a20f-fa1e17af02dc_8
```

docker exec -it 74b149443cc8 bash

进入到容器查看

ls /usr/share/nginx/html

```
[root@nod1 ~]# docker exec -it 74b149443cc8 bash
root@74b149443cc8:/# ls /usr/share/nginx/html
ceshi.txt
root@74b149443cc8:/#
```

自定义网络运行服务

docker network create --driver overlay mynetwork

创建 overlay 类型网络 mynetwork

```
[root@master ~]# docker network create --driver overlay mynetwork
45eifizpdsg3fh8anr6uw0bq3
[root@master ~]#
```

docker service create --replicas 3 --network mynetwork --name myweb nginx

使用自定义网络 mynetwork 新建容器 myweb

```
[root@master ~]# docker service create  --replicas 3 --network mynetwork --name myweb nginx
zg04xa2vnxm8uruk348m30924
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
[root@master ~]#
```

docker service ps myweb

```
[root@master ~]# docker service  ps myweb
ID             NAME      IMAGE          NODE      DESIRED STATE   CURRENT STATE            ERROR     PORTS
ipx8rhxuo4wc   myweb.1   nginx:latest   nod2      Running         Running about a minute ago
jxf9bmh8item   myweb.2   nginx:latest   nod1      Running         Running about a minute ago
sk167atejb7n   myweb.3   nginx:latest   nod2      Running         Running about a minute ago
[root@master ~]#
```

十五、Docker （weave scope）监控管理

（一）docker 监控命令

1.docker ps

docker ps

```
[root@centos7test ~]# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS     NAMES
6d2dcd286021   centos:7                       "/bin/bash"              42 minutes ago   Up 42 minutes             gallant_elion
90d5f500e7d0   centos:7                       "/bin/bash"              42 minutes ago   Up 42 minutes             nostalgic_golick
54965793b3e6   centos:7                       "/bin/bash"              42 minutes ago   Up 42 minutes             objective_euler
efa91e164b99   centos:7                       "/bin/bash"              42 minutes ago   Up 42 minutes             quirky_mcclintock
5812c0fa57d8   weaveworks/scope:latest        "/home/weave/entrypo…"   51 minutes ago   Up 51 minutes             weavescope
[root@centos7test ~]#
```

2. docker top

docker top 6d2dcd286021

```
[root@centos7test ~]# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS     NAMES
6d2dcd286021   centos:7                       "/bin/bash"              48 minutes ago   Up 48 minutes             gallant_elion
90d5f500e7d0   centos:7                       "/bin/bash"              48 minutes ago   Up 48 minutes             nostalgic_golick
54965793b3e6   centos:7                       "/bin/bash"              48 minutes ago   Up 48 minutes             objective_euler
efa91e164b99   centos:7                       "/bin/bash"              48 minutes ago   Up 48 minutes             quirky_mcclintock
5812c0fa57d8   weaveworks/scope:latest        "/home/weave/entrypo…"   57 minutes ago   Up 57 minutes             weavescope
[root@centos7test ~]# docker top 6d2dcd286021
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                4197                4175                0                   04:08               pts/0               00:00:00            /bin/bash
[root@centos7test ~]#
```

3. docker stats

docker stats

```
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O       BLOCK I/O       PIDS
6d2dcd286021   gallant_elion     0.00%     392KiB / 3.686GiB     0.01%     648B / 0B     0B / 0B         1
90d5f500e7d0   nostalgic_golick  0.00%     396KiB / 3.686GiB     0.01%     648B / 0B     0B / 0B         1
54965793b3e6   objective_euler   0.00%     388KiB / 3.686GiB     0.01%     648B / 0B     0B / 0B         1
efa91e164b99   quirky_mcclintock 0.00%     1.949MiB / 3.686GiB   0.05%     648B / 0B     9.71MB / 0B     1
5812c0fa57d8   weavescope        12.12%    136.1MiB / 3.686GiB   3.61%     0B / 0B       0B / 0B         29
```

（二）sysdig

1.下载 sysdig

docker pull sysdig/sysdig:0.27.1

2.

```
# docker container run -it --rm --name=sysdig --privileged=true \
--volume=/var/run/docker.sock:/host/var/run/docker.sock \
--volume=/dev:/host/dev \
--volume=/proc:/host/proc:ro \
```

--volume=/boot:/host/boot:ro \

--volume=/lib/modules:/host/lib/modules:ro \

--volume=/usr:/host/usr:ro \

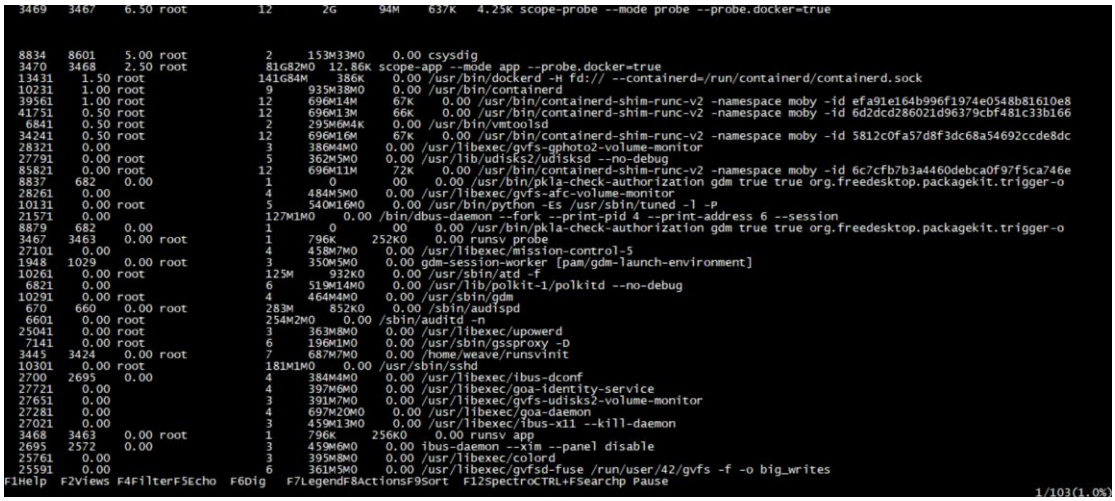
sysdig/sysdig:0.27.1

或者

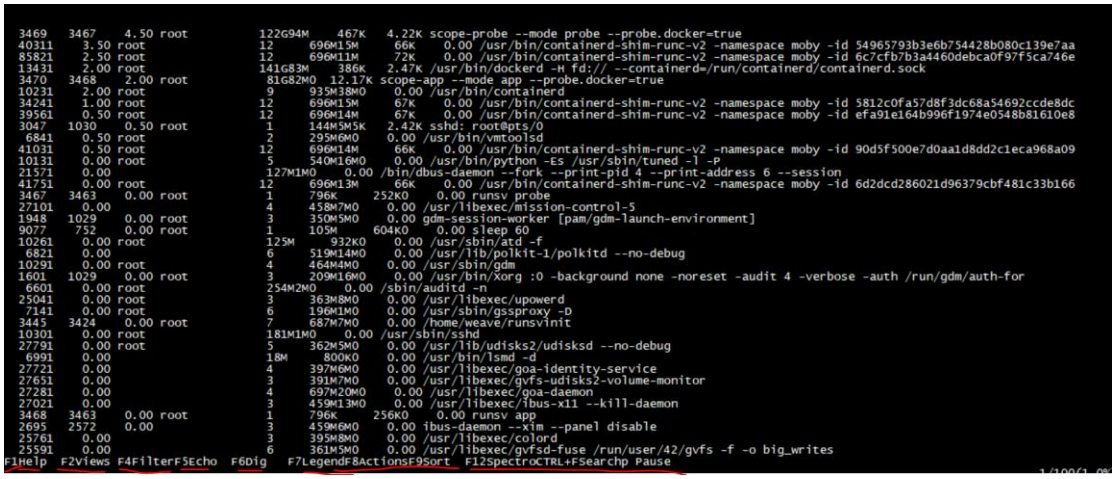
docker container run -it --rm --name=sysdig --privileged=true --volume=/var/run/docker.sock:/host/var/run/docker.sock --volume=/dev:/host/dev --volume=/proc:/host/proc:ro --volume=/boot:/host/boot:ro --volume=/lib/modules:/host/lib/modules:ro --volume=/usr:/host/usr:ro sysdig/sysdig:0.27.1

csysdig

Sysdig 开始运行



sysdig 的界面类似 linux 的 top，但是功能更强大，点击底部的 View 或者按 F2 可以看到很多监控分类，涵盖了操作系统的各个方面. sysdig 显示的是实时数据，看不到变化和趋势。



（三）weave scope

1.下载

curl -L git.io/scope -o /usr/local/bin/scope

2.赋予脚本执行权限

chmod a+x /usr/local/bin/scope

3.执行脚本

cd /usr/local/bin/scope

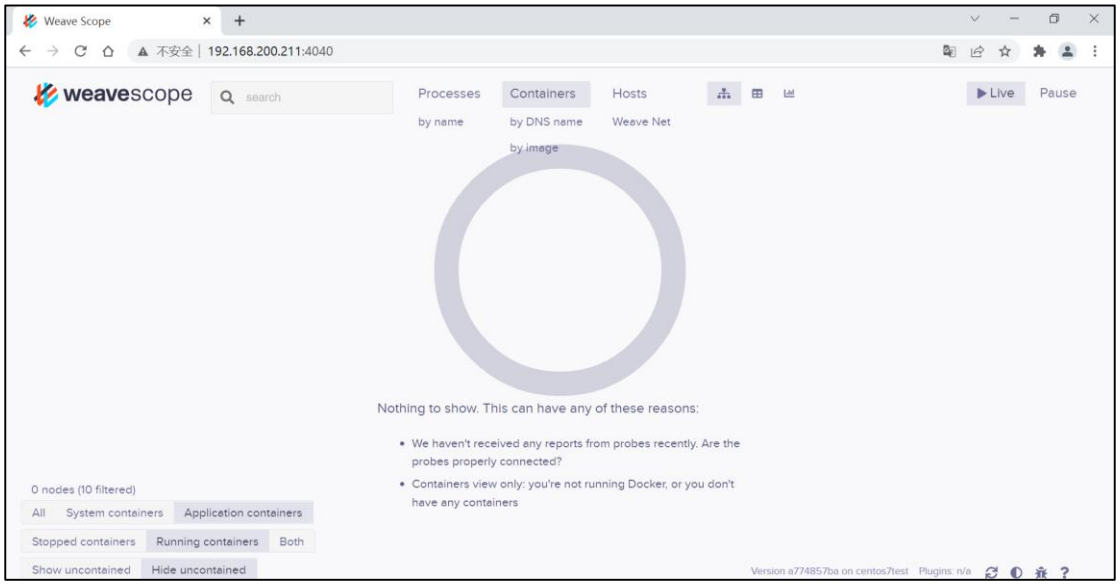
./scope launch

在下图看到 weave scope 登录网址

```
[root@centos7test ~]# cd /usr/local/bin/scope
[root@centos7test scope]# ./scope launch
Unable to find image 'weaveworks/scope:latest' locally
latest: Pulling from weaveworks/scope
ba3557a56b15: Pull complete
c13051a275bd: Pull complete
3ed215655c93: Pull complete
6927287a47e7: Pull complete
26f0e0cb9bd7: Pull complete
ee10628da55f: Pull complete
1f15fa4f3644: Pull complete
b3b44852124e: Pull complete
36f6d45edc34: Pull complete
Digest: sha256:064dd36606a9d925e543aaa2340efe290b4ce8bb906a2a3422de93c2cbd4b986
Status: Downloaded newer image for weaveworks/scope:latest
5812c0fa57d8f3dc68a54692ccde8dcd3bab4c7778dc724f785731a8af2cb04a
Scope probe started
Weave Scope is listening at the following URL(s):
  * http://192.168.122.1:4040/
  * http://192.168.200.211:4040/
[root@centos7test scope]#
```

4.登录到 weave scope

http://192.168.200.211:4040/



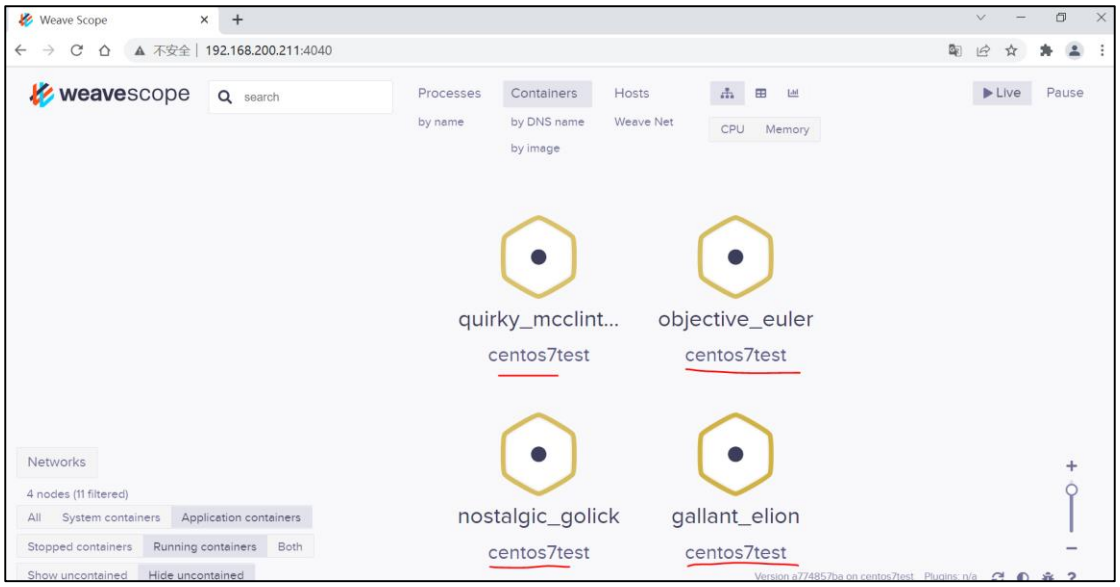
5.查看运行的脚本

运行 4 个 centos:7

docker run -it -d centos:7

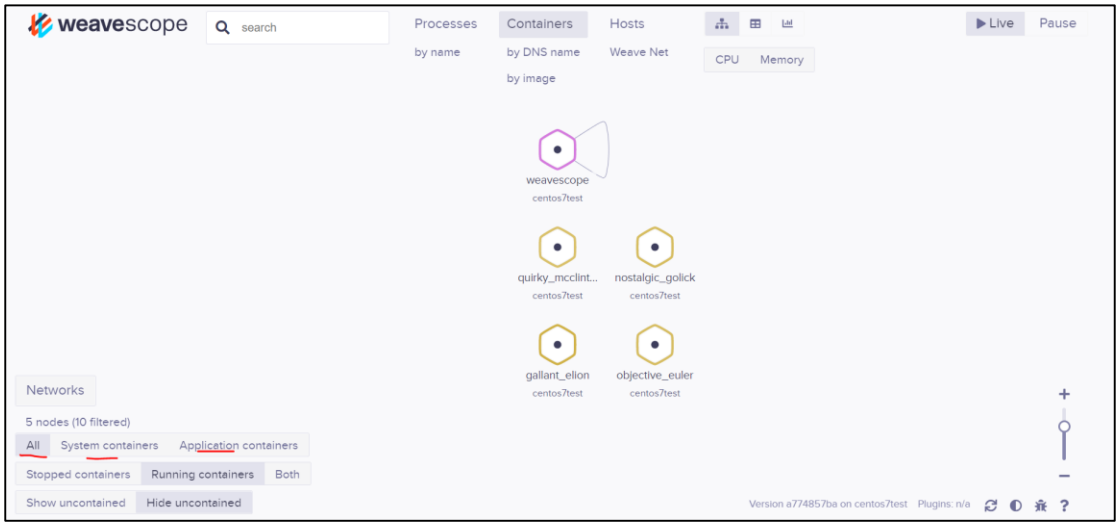
```
[root@centos7test scope]# docker run -it -d centos:7
efa91e164b996f1974e0548b81610e87da7e1323e77e54bfc210d5a34c5b8811
[root@centos7test scope]# docker run -it -d centos:7
54965793b3e6b754428b080c139e7aad8a1341132308a9ba31efe05ca0a73cbd
[root@centos7test scope]# docker run -it -d centos:7
90d5f500e7d0aa1d8dd2c1eca968a09063b628a9721a80365fd0a4943c256587
[root@centos7test scope]# docker run -it -d centos:7
6d2dcd286021d96379cbf481c33b1669104e3bfd8ee9a084b02fed897d7a690a
[root@centos7test scope]#
```

再次看，可以看到运行的容器



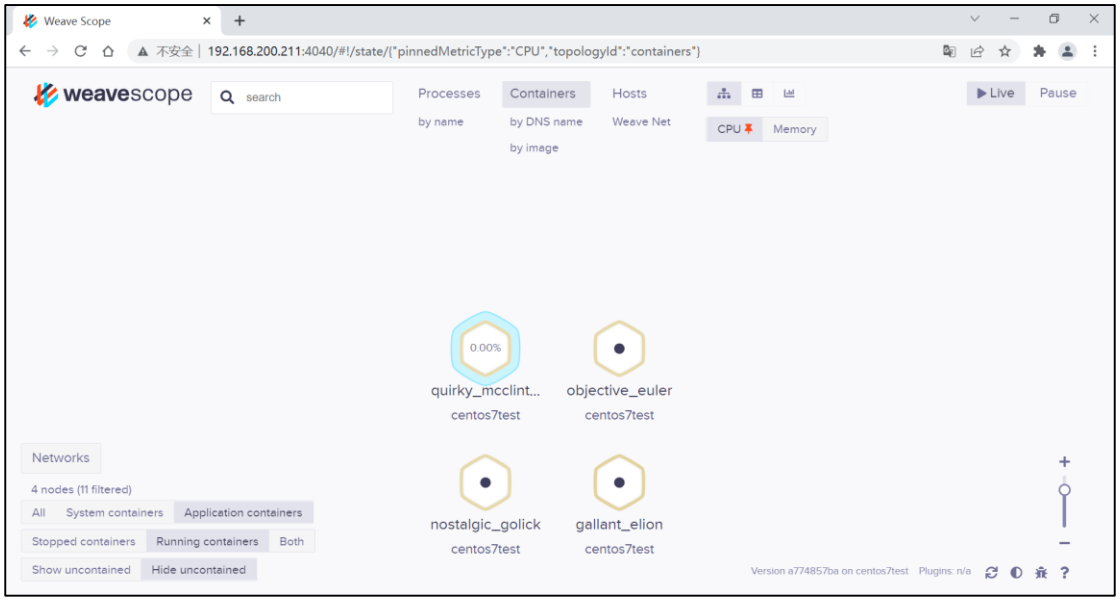
6.查看运行容器

按照下图所示可以查看不同的运行容器

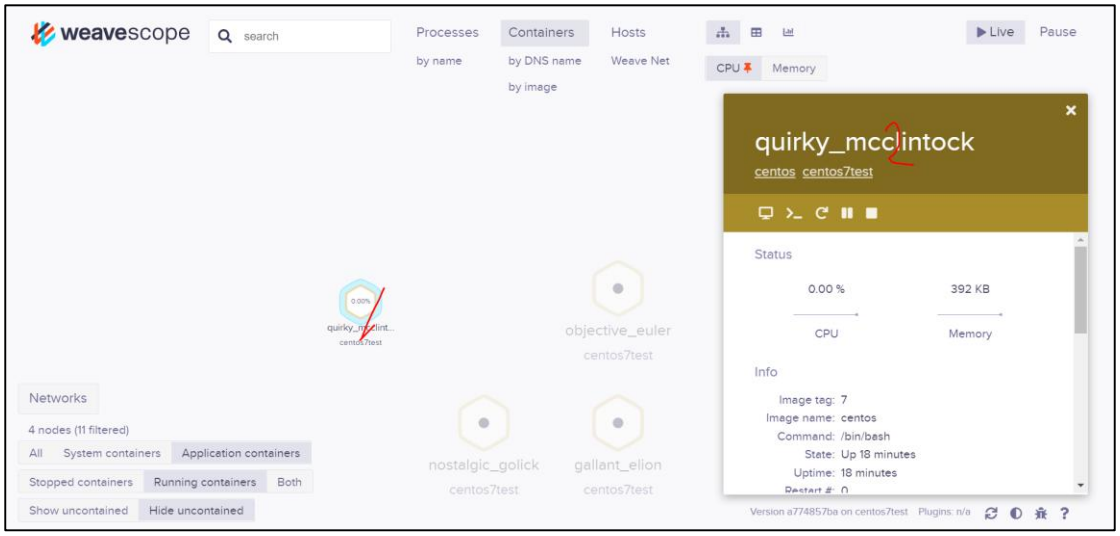


7.查看资源占用

例如查看 cpu，单击右上角 cpu，然后把鼠标移动到运行容器会以百分比显示

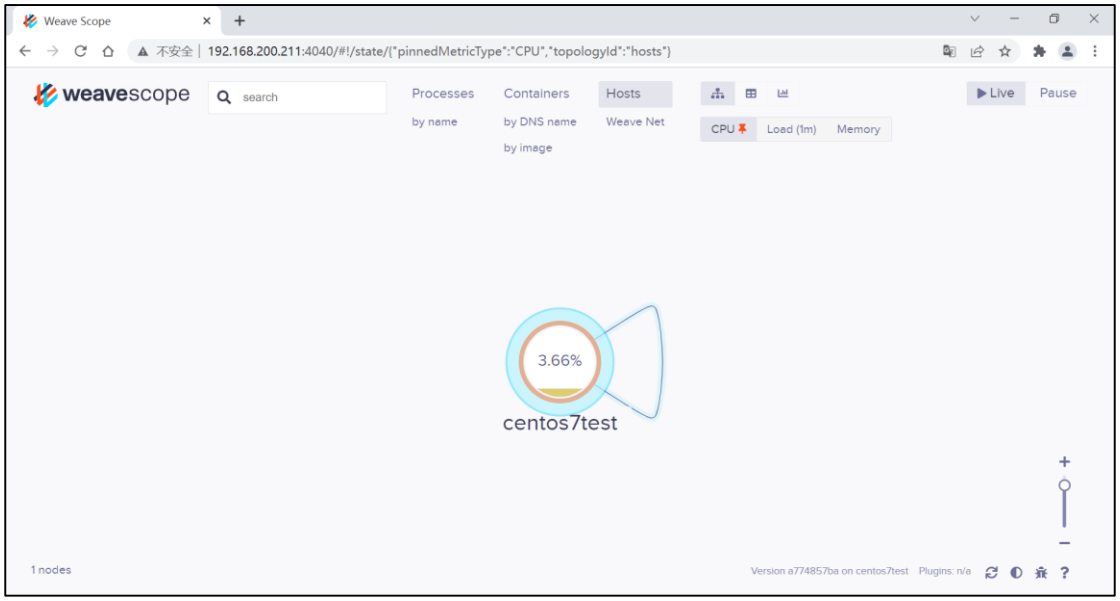


8.具体查看容器信息，方法：单击具体容器图标



9.查看宿主机

单击右上角“hosts”，鼠标移到图标上就可以具体查看了



10.多宿主机监控

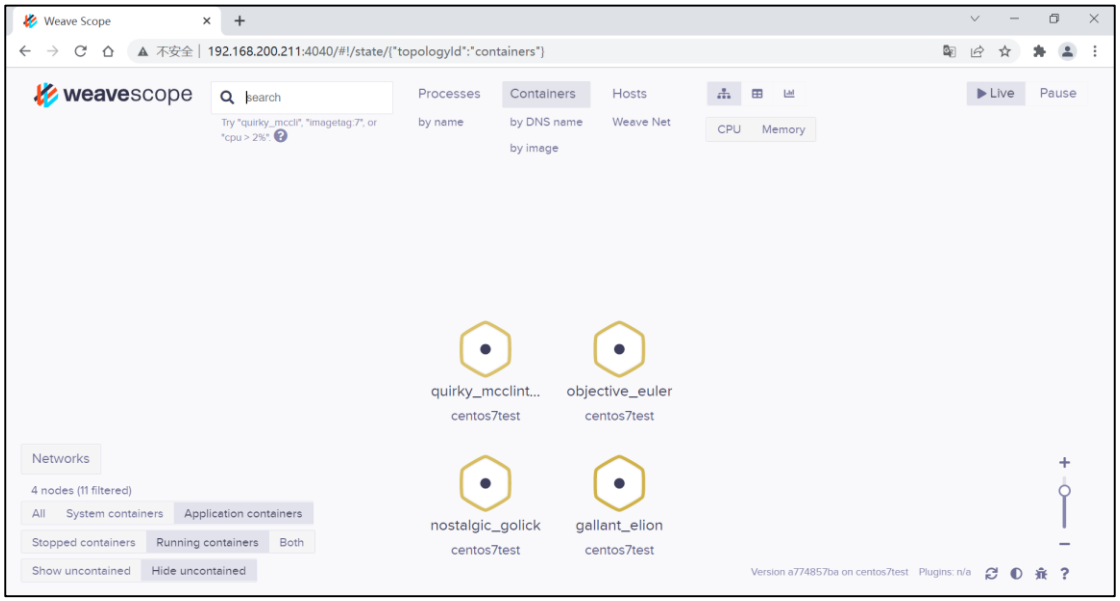
方法：

（1）分别在宿主上安装 weave scope

（2）分别在宿主上运行

.# ./scope launch 宿主机 1ip 宿主机 2ip

11.搜索



十六、Docker 专业基础

（一）ssh 登录链接 centos

1.安装 openssh-server

yum install -y openssh-server

2./etc/ssh/ 目录下的 sshd 服务配置文件 sshd_config，用 Vim 编辑器打开

将文件中，关于监听端口、监听地址前的 # 号去除

```
Port 22
#AddressFamily any
ListenAddress 0.0.0.0
ListenAddress ::

HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
```

```
    #AuthorizedPrincipalsFile none

    #AuthorizedKeysCommand none
    #AuthorizedKeysCommandUser nobody

    # For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
    #HostbasedAuthentication no
    # Change to yes if you don't trust ~/.ssh/known_hosts for
    # HostbasedAuthentication
    #IgnoreUserKnownHosts no
    # Don't read the user's ~/.rhosts and ~/.shosts files
    #IgnoreRhosts yes

    # To disable tunneled clear text passwords, change to no here!
    #PasswordAuthentication yes
    #PermitEmptyPasswords no
    PasswordAuthentication yes

    -- INSERT --
```

开启 sshd 服务

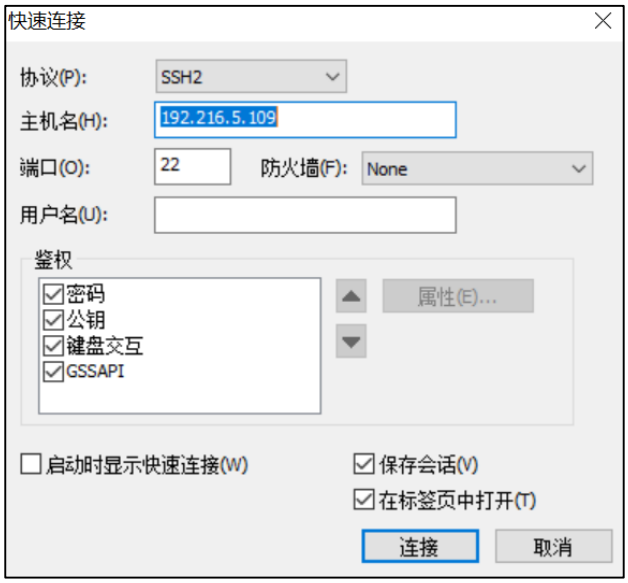
systemctl start sshd

开机运行 sshd 服务

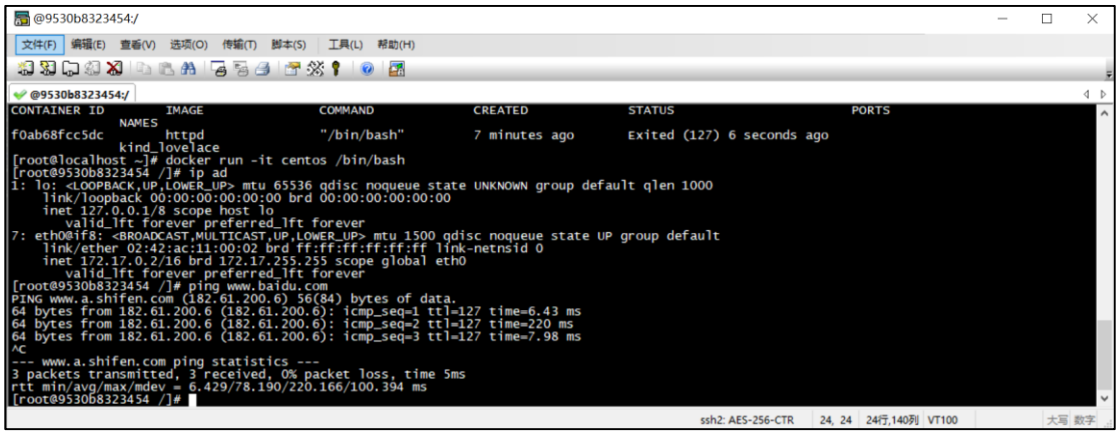
systemctl enable sshd

3.ssh 客户端登入（securecrt）

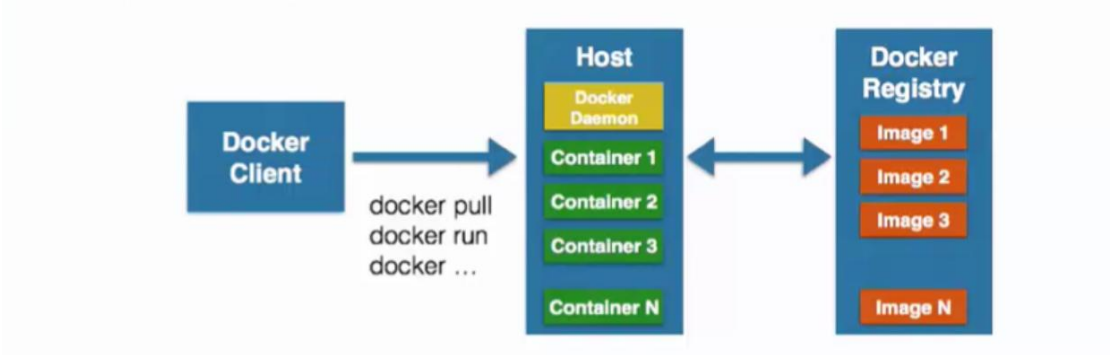
（1）

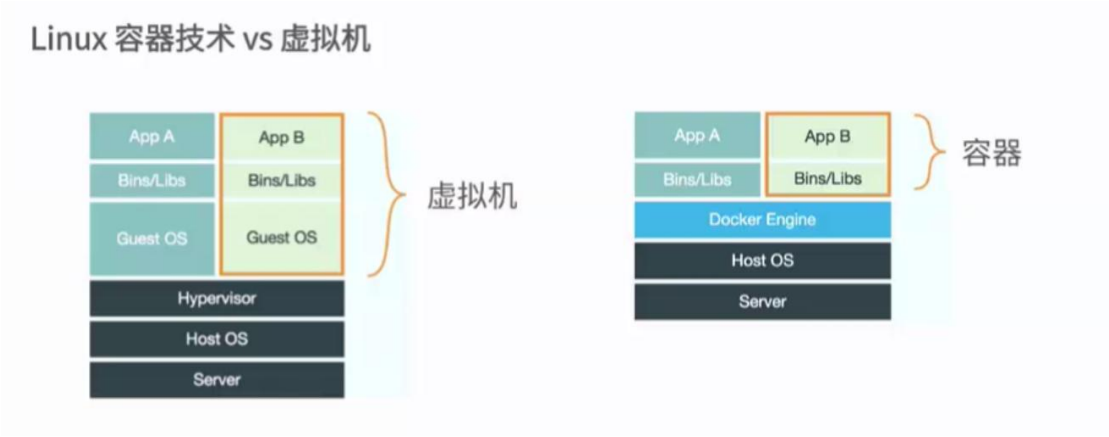


（2）



（二）docker 的基本组成





（三）Dockerfile 构建过程

1.基础知识
（1）每条保留字指令都必须为大写字母且后面要跟随至少一个参数
（2）指令按照从上到下,顺序执行
（3）# 表示注释
（4）每条指令都会创建一个新的镜像层,并对镜像进行提交
2.docker 执行 dockerfile 的大致流程
第一步,docker 从基础镜像运行一个容器
第二步,执行一条指令并对容器做出修改
第三步,执行类似 docker commit 的操作提交一个新的镜像层
第四步,docker 再基于刚提交的镜像运行一个新容器
第五步,执行 dockerfile 中的下一条指令知道所有指令都执行完成
3.docker 应用软件角度
dockerfile 是软件的原材料
docker 镜像是软件的交付品
docker 容器是软件的运行态
dockerfile 面向开发
docker 镜像成为交付标准
docker 容器则涉及部署与运维
4.Dockerfile 是用来构建 Docker 镜像的构建文件，是由一系列命令和参数构成的脚本。
构建三步骤
编写 Dockerfile 文件→docker build→docker run

（四）Dockerfile 指令

1、基础指令：FROM、MAINTAINER、RUN、EXPOSE
（1）~ FROM
指定镜像继承自哪个基础镜像（基础镜像相当于 java 中的基础类），第一条非注释指令必须

为 FROM 指令；通过 from 指令指定的镜像名必须是已经存在的镜像，后续指令都会基于这个镜像来执行；

格式为 FROM <image> 或 FROM <image>:<tag> ；

（2）~ MAINTAINER

指定维护者信息；

格式为 MAINTAINER <name>；

（3）~ RUN

指定在镜像构建过程中要执行的命令行命令；

有 2 中格式：

shell 格式：RUN <command>

将在 **shell 终端**中运行命令，就像直接在命令行中输入的命令一样；

RUN echo 'zxj' > /usr/local/docker/tomcat/html/index.html 等价于直接在终端输入 echo 命令；

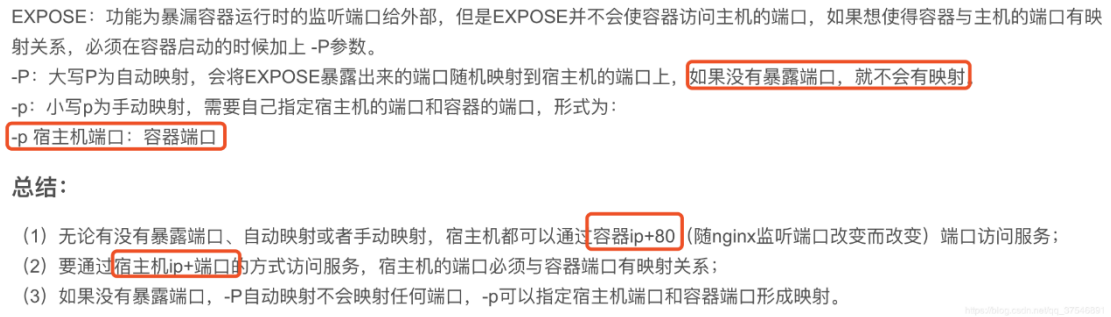
exec 格式：RUN ["executable", "param1", "param2"]

使用 **exec** 执行，更像是函数调用中的格式；指定使用其它终端可以通过第二种方式实现，例如 RUN [“/bin/bash”, “-c”,”echo hello”] ；

（4）~ EXPOSE

指定容器要打开的端口，这个端口与宿主机的端口进行映射；（指定运行该镜像所使用的容器的端口，可以指定一个或多个端口，也可以使用多个这个命令）；

格式为： EXPOSE <port> [<port>...]；



2、CMD、ENTRYPOINT

这两个指令用来指定在容器运行时运行的命令；

（1）~ CMD

Docker 不是虚拟机，容器就是进程；既然是进程，那么在启动容器的时候，需要指定所运行的程序及参数；CMD 指令就是用于指定默认的容器主进程的启动命令的；相当于 cmd 下输入 Tomcat 的 start.bash；

指令用于指定一个容器的默认的可执行体，也就是容器启动后默认执行的命令；

若 docker run 没有指定任何执行命令，或 Dockerfile 里面没有 ENTRYPOINT 指令，那么就会使用 CMD 指定的默认的执行命令执行；否则就会使用 docker run 后面的命令或 ENTRYPOINT 指令的参数覆盖 CMD 参数；

有点类似于 RUN 命令，只是 RUN 命令是指定**镜像被构建时**要运行的命令，而 CMD 是指定**容器被启动时**要运行的命令；

这和使用 docker run 命令启动容器时指定要运行的命令很类似：docker run -i -t 仓库名 /bin/bash<= 等价于 =>CMD ["/bin/bash"]；也可以为要运行的命令指定参数：CMD ["/bin/bash","-l"]；

Dockerfile 只能有一条 CMD：如果指定了多条命令，只有最后一条会被执行；若想在启动容器时运行多个进程或多条命令，可以考虑使用类似 Supervisor 这样的服务管理工具；

docker run **命令行覆盖 CMD**：如果用户启动容器（docker run）的时候指定了要运行的命令，则命令行中指定的命令会覆盖掉在 Dockerfile 里 CMD 指定的命令；

支持三种格式：

CMD ["executable","param1","param2"] ：要运行的命令是存放在一个数组结构中的；使用 exec 执行，推荐方式；（命令在没有终端的环境中使用此方式执行） 第一个参数必须是命令的全路径；

CMD command param1 param2 在 /bin/bash 中执行，提供给需要交互的应用；

CMD ["param1","param2"] 用于给 ENTRYPOINT 提供参数；

（2）~ ENTRYPOINT

entrypoint 是容器的入口；是真正的的用于定义容器启动以后的执行体的；

配置**容器启动后**执行的命令；

没有指定 ENTRYPOINT 时，就用 CMD 指定的命令+参数来启动容器；若指定了 ENTRYPOINT，CMD 指定的字符串就会变成 ENTRYPOINT 指令的参数；两者可以组合使用；

若在 docker run 命令行中指定了-g 参数，则-g 后面的参数会覆盖 cmd 参数传递给 ENTRYPOINT，若要使 run 覆盖 ENTRYPOINT，需要在 run 后面加上–entrypoint 参数；

So 可以构建一个镜像，该镜像既可以运行一个默认的命令(CMD 指定的)，同时也支持通过 docker run 命令行为该命令指定可覆盖的选项或标志；

ENTRYPOINT 不容易被 docker run 提供的参数覆盖，若想被 docker run 指定的命令覆盖，则需要在 docker run 命令中指定--entrypoint 选项。

每个 Dockerfile 中只能有一个 ENTRYPOINT ，当指定多个时，只有最后一个起效；

两种格式：

ENTRYPOINT ["executable", "param1", "param2"] ：通过数组的方式为命令指定相应的参数，参数可有可不有；

ENTRYPOINT command param1 param2 （shell 中执行）

一般使用 entrypoint 的中括号形式作为 docker 容器启动以后的默认执行命令，里面放的是不变的部分，可变部分比如命令参数可以使用 cmd 的形式提供默认版本，也就是 run 里面没有任何参数时使用的默认参数。如果我们想用默认参数，就直接 run，否则想用其他参数，就 run 里面加参数；

ENTRYPOINT 和 **CMD** 的**区别**：

ENTRYPOINT 指定了该镜像启动时的入口，CMD 则指定了容器启动时的命令，当两者共用时，完整的启动命令像是 ENTRYPOINT + CMD 这样；使用 ENTRYPOINT 的好处是在我们启动镜像就像是启动了一个可执行程序，在 CMD 上仅需要指定参数；另外在我们需要自定义 CMD 时不容易出错；

>> 3、WORKDIR、ENV、USER

这三个指令用来指定镜像在构建及运行时的环境设置；

（1）~ WORKDIR

使用 WORKDIR 指令可以来指定工作目录(或者称为当前目录)，将工作目录换到指定路径，以后各层的当前目录就被改为指定的目录，如该目录不存在，WORKDIR 会帮你建立目录；相当于 **cd**；但是不要在 Dockerfile 中 使用 RUN cd /usr/..../进行目录的切换；

RUN cd /app

RUN echo "hello" > world.txt

如果将这个 Dockerfile 进行构建镜像运行后，会发现找不到 /app/world.txt 文件，或者其内容不是 hello；原因其实很简单，在 Shell 中，连续两行是同一个进程执行环境，因此前一

个命令修改的内存状态，会直接影响后一个命令；而在 Dockerfile 中，这两行 RUN 命令的执行环境根本不同，是两个完全不同的容器，这就是对 Dockerfile 构建分层存储的概念不了解所导致的错误；

每一个 RUN 都是启动一个容器、执行命令、然后提交存储层文件变更；第一层 RUN cd /app 的执行仅仅是当前进程的工作目录变更，一个内存上的变化而已，其结果不会造成任何文件变更；而到第二层的时候，启动的是一个全新的容器，跟第一层的容器更完全没关系，自然不可能继承前一层构建过程中的内存变化；

因此如果需要改变以后各层的工作目录的位置，那么应该使用 WORKDIR 指令；

我们可以使用该指令为 Dockerfile 中后续的 RUN 、 CMD 、 ENTRYPOINT 等一系类指令设置工作目录，也可以为最终容器设置工作目录；Eg：

```
WORKDIR /opt/webapp/db
RUN bundle install
WORKDIR /opt/webapp
ENTRYPOINT [ "rackup" ]
```

将工作目录切换为/opt/webapp/db

之后运行这个命令

之后又将工作目录设置为/opt/webapp

最后设置了ENTRYPOINT指令来启动rackup，命令

可以通过-w 标志在运行时覆盖工作目录：docker run -it -w /var/log ubuntu pwd：该命令会将容器内部的工作目录设置为/var/log；

格式为 WORKDIR /path/to/workdir；

可以使用多个 WORKDIR 指令，后续命令如果参数是相对路径，则会基于之前命令指定的路径；例如：

WORKDIR /a

WORKDIR b

WORKDIR c

RUN pwd

则最终路径为 /a/b/c ；

（2）~ ENV

指令用来在镜像构建过程中设置环境变量，为容器里面的环境设置变量；

格式为 ENV <key> <value> ； Eg： ENV PATH /usr/local/nginx/sbin:$PATH

指定一个环境变量，这个新的环境变量可以在后续的任何 RUN 指令中使用，并在容器运行时保持，这就如同在命令前面指定了环境变量前缀一样；

也可以在其他指令中直接使用这些环境变量：ENV TARGET_DIR /opt/app、WORKDIR $TARGET_DIR；

如果需要，可以通过在环境变量前加上一个反斜杠来进行转义；

ENV 设置的环境变量会被持久保存到从镜像创建的任何容器中；

也可以使用 docker run 命令行的-e 标志来传递环境变量，但是这个变量只会在运行时有效；

（3）~ USER

指令用来指定该镜像会以什么样的用户去执行；

Eg：USER nginx：基于该镜像启动的容器会以 nginx 用户的身份来运行；

格式为 **USER daemon**；例如： RUN useradd -s /sbin/nologin -M www；

USER：指定执行该命令的用户；可以指定运行容器时的用户名或 UID，以及组或 GID，甚至是两者的结合，后续的 RUN 也会使用指定用户。

```
USER user
USER user:group
USER uid
USER uid:gid
USER user:gid
USER uid:group
```

当服务不需要管理员权限时，可以通过该命令指定运行用户。并且可以在之前创建所需要的用户；
可以在 docker run 命令中通过-u 选项来覆盖该指令的值；
若不通过 USER 指令指定用户，默认用户是 **root**；

>> 4、VOLUMN、COPY、ADD
这三个指令用来设置镜像的目录和文件；
（1）~ VOLUMN
指令用来向基于镜像创建的**容器**添加数据卷；**mount point**；**挂载目录**；创建一个可以从本地主机或其他容器挂载的挂载点，一般用来存放数据库和需要保持的数据等；
格式为 VOLUME ["/opt/project"]：这条指令将会为基于此镜像创建的任何容器 创建一个名为 /opt/project 的挂载点；
也可以通过制定数组的方式指定多个卷：VOLUMN ["/opt/project","/data"]
（2）~ COPY
ADD <源路径> <目标路径>
COPY 指令将从构建上下文目录中 <源路径> 指定的文件/目录，复制到新的一层的镜像内的 <目标路径> 位置；若源路径是一个目录，那么整个目录都将被复制到容器中，包括文件系统元数据；若源文件为任何类型的文件，则该文件会随同元数据一起被复制；（以 **/** 结尾的路径，Docker 会将它作为目录进行复制；）
指令非常类似于 ADD，它们根本的不同是 COPY 只关心在构建上下文中复制本地文件，而不会去做文件提取和解压的工作；
源路径：相对于构建上下文的相对路径；
必须是一个与当前构建环境相对的文件或目录，本地文件都放到和 Dockerfile 同一个目录下；
不能复制该目录之外的任何文件，因为构建环境会上传到 Docker 守护进程，而复制是在 Docker 守护进程中进行的；**任何位于构建环境之外的东西都是不可用的！**
源路径可以是多个，也可以是通配符；
目标路径：镜像中文件夹的位置；
COPY 的**目标位置** 可以是容器内的绝对路径，也可以是相对于工作目录的相对路径（工作目录可以用 WORKDIR 指令来指定）；
目标路径不需要事先创建，如果目录不存在会在复制文件前先行创建缺失目录；
任何由该指令创建的文件或者目录的 UID 和 GID 都会设置为 0；
使用 COPY 指令，源文件的各种元数据都会保留，比如读、写、执行权限、文件变更时间等；
这个特性对于镜像定制很有用，特别是构建相关文件都在使用 Git 进行管理的时候；
（3）~ ADD
ADD <源文件位置> <目的文件位置>
指令用来将构建环境下的 文件/目录 复制到镜像中；也就是 往容器里面添加文件；相当于 COPY，但是比 COPY 功能更强大；

源路径 ：

可以是 Dockerfile 所在目录的一个相对路径；

可以是一个 URL：可以将远程文件加入到容器里面；

Docker 引擎会试图去下载这个链接的文件放到 <目标路径> 去，下载后的文件权限自动设置为 600，如果这并不是想要的权限，那么还需要增加额外的一层 RUN 进行权限调整；

另外，如果下载的是个压缩包，需要解压缩，也一样还需要额外的一层 RUN 指令进行解压缩；所以不如直接使用 RUN 指令，然后使用 wget 或者 curl 工具下载，处理权限、解压缩、然后清理无用文件更合理；

可以是一个 tar 文件：压缩格式为 gzip, bzip2 以及 xz 的情况下，复制进容器会自动解压这个压缩文件到<目标路径>；

在某些情况下，这个自动解压缩的功能非常有用，比如官方镜像 ubuntu 中：ADD ubuntu-xenial-core-cloudimg-amd64-root.tar.gz /

但在某些情况下，如果我们真的是希望复制个压缩文件进去，而不解压缩，这时就不可以使用 ADD 命令了；

目标路径：

若目的位置不存在，Docker 会创建这个全路径，包括路径中的任何目录；新创建的目录和文件的模式为 0755，并且 UID 和 GID 都是 0；

ADD 指令会使得**构建缓存失效**，从而可能会令镜像构建变得比较缓慢：若通过 ADD 指令向镜像添加一个文件或者目录，那么将是 Dockerfile 中的后续指令都不能继续使用之前的构建缓存；

5、ONBUILD

为镜像添加触发器 trigger；

ONBUILD：为镜像添加触发器 trigger，当一个镜像被用作其他镜像的基础镜像时，这个触发器将会被执行；配置 当所创建的镜像 作为其它新创建镜像的基础镜像时，所执行的操作指令；

格式为 ONBUILD [INSTRUCTION]；

在构建本镜像时不生效，在基于此镜像构建镜像时生效；当子镜像在构建时，会插入触发器中的指令；ONBUILD 触发器会按照在父镜像中指定的顺序执行，并且只被执行一次，也就是说只能在子镜像中执行，而不会在孙子镜像中执行；

触发器会在构建过程中插入新指令，我们可以认为这些指令是紧跟在 **FROM 之后**指定的；

触发器可以是任何构建指令：ONBUILD ADD . /app/src、ONBUILD RUN cd /app/src && make，代码将会在创建的镜像中加入 ONBUILD 触发器，ONBUILD 指令可以在镜像上运行 docker inspect 命令来查看；

```
$ sudo docker inspect 508efa4e4bf8
...
"OnBuild": [
    "ADD . /app/src",
    "RUN cd /app/src/ && make"
]
...
```

可以将含有 ONBUILD 指令的 Dockerfile 文件作为一个**通用的 Web 应用程序的模板**，可以基于这个模板来构建 Web 应用程序：

首先创建一个含有 ONBUILD 指令的 Dockerfile 文件，并以此来构建全新镜像 A，然后再构建

一个名为 B 的镜像 FROM A（以镜像 A 为基础镜像），构建镜像 B 时，先执行 DockerfileB 的 FROM 指令，之后执行基础镜像 A 的 Dockerfile 文件中的 ONBUILD 指令，然后才会继续执行 DockerfileB 中的第二条指令及后续各指令；
ONBUILD 指令中不能使用的指令：FROM、MAINTAINER、ONBUILD；（防止在 Dockerfile 构建过程中产生递归调用的问题）

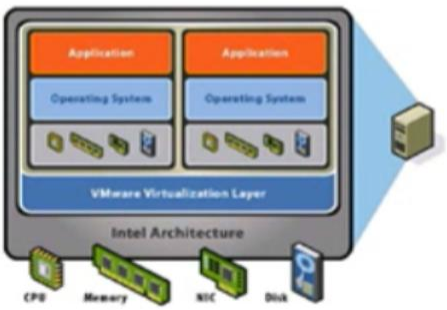
（五）虚拟化与容器技术

传统虚拟化技术

- 纯软件的虚拟化是通过对于硬件层的模拟从而实现允许运行多个操作系统
- 硬件辅助虚拟化需要硬件层面对于虚拟化的支持，类似Interl-VT技术等，具有更高的运行效率

传统虚拟化

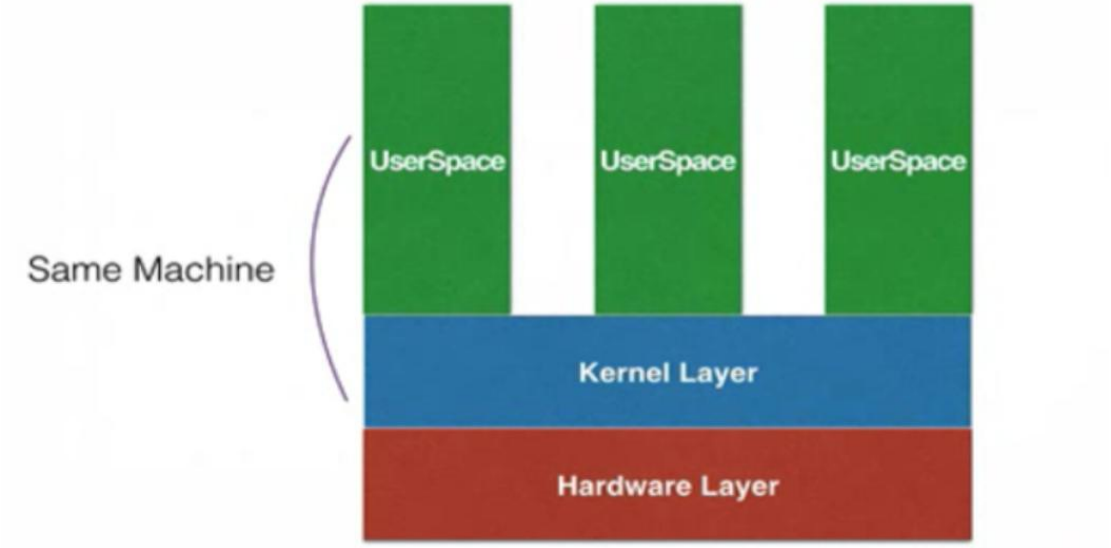
- Guets OS 运行在VMM之上
- VMM (Virtual Machine Monitor) 运行在Host OS之上
- Host OS 负责真正的对于底层硬件的调用



解决方案

- 商业解决方案
  - VMware vSphere，VMware Fusion
- 开源解决方案
  - OpenStack,KVM

System-level虚拟化



解决方案

- LXC
- OpenVZ
- **Docker**
- lmctfy，google开源的容器虚拟化的实现

传统虚拟化与容器虚拟化的区别

	传统虚拟化	容器虚拟化
创建速度	很慢	非常快
性能影响	通过对于硬件层的模拟，增加了系统调用链路的环节，有性能损耗	共享Kernel，几乎没有性能损耗
资源消耗	很大	很小，一台机器可以轻松创建多个Container
操作系统覆盖	支持Linux，Windows，Mac等	仅仅Kernel所支持的OS

新运维

https://www.unixhot.com/

cat /proc/cpuinfo 查看 cpu 核心数

```
[root@localhost ~]# cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 142
model name      : Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
stepping        : 10
microcode       : 0xca
cpu MHz         : 1992.004
cache size      : 8192 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat ps
tscp lm constant_tsc arch_perfmon nopl xtopology tsc_reliable nonstop_tsc eagerfpu pni
vbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowpr
c_adjust bmi1 avx2 smep bmi2 invpcid mpx rdseed adx smap clflushopt xsaveopt xsavec ar
ilities
```

附录、Docker 常用镜像运行

（一）httpd 运行

docker run -d -p 80:80 httpd

（二）nginx 运行

docker run -d -p 80:80 nginx

（三）centos 运行

docker run -it -d centos

```
[root@localhost ~]# docker run  -it -d  centos
093a27ec55e6e611778f2a3c7de570c39aa1ec2fe9c7a01d0106edc6eb3d1fd0
```

docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
093a27ec55e6        centos              "/bin/bash"         2 seconds ago       Up 1 second                             practical_blackwell
```

docker exec -it 093a27ec55e6 bash

```
[root@localhost ~]# docker exec -it 093a27ec55e6  bash
[root@093a27ec55e6 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
32: eth0@if33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
[root@093a27ec55e6 /]#
```

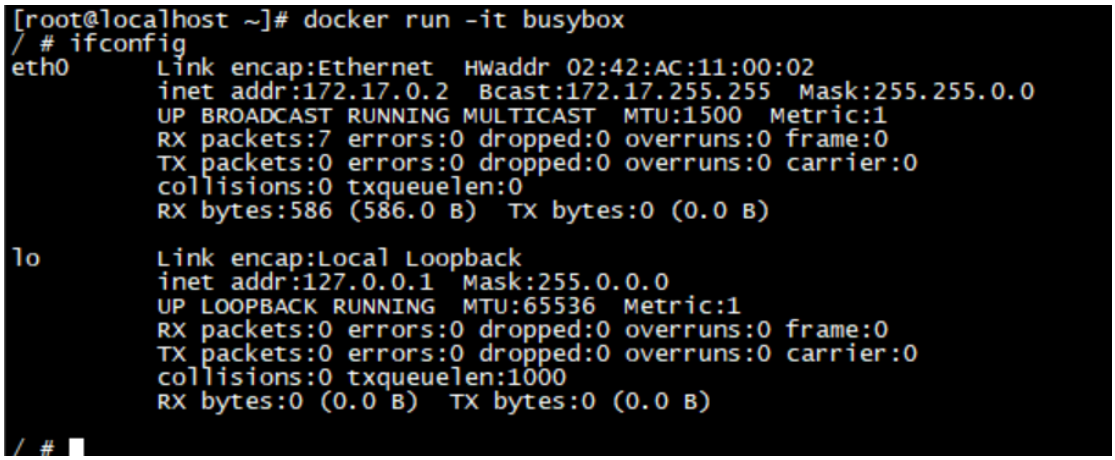
（四）hello-world 运行

docker run hello-world

```
[root@localhost _data]# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

（五）busybox 运行

docker run -it busybox



（六）docker 镜像网站

https://hub.docker.com/

配置阿里云为 yum 下载源

yum install -y wget

rm -rf /etc/yum.repos.d/*

wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo