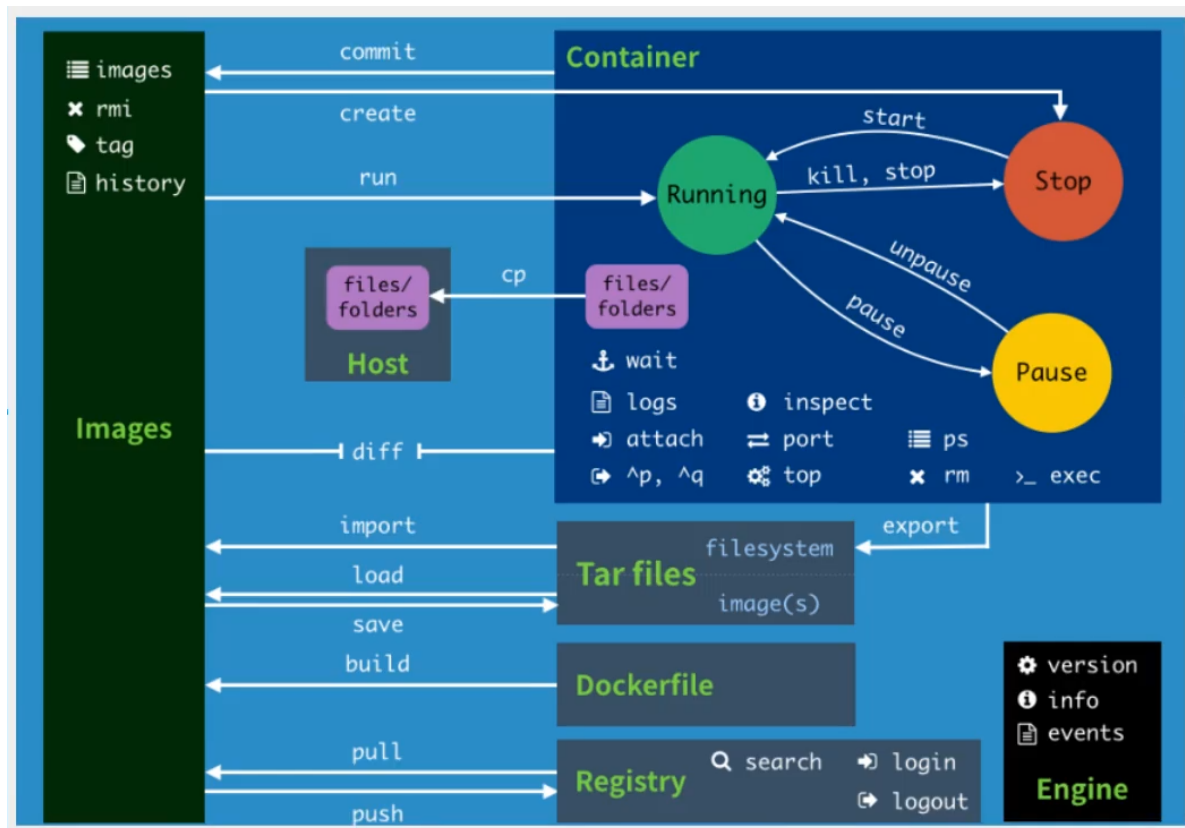


# Docker

DockerHub: <https://hub.docker.com/>

帮助文档地址: <https://docs.docker.com/>



## Docker安装

### 卸载旧版本

```
yum remove docker \
    docker-client \
    docker-client-latest \
    docker-common \
    docker-latest \
    docker-latest-logrotate \
    docker-logrotate \
    docker-engine
```

### 需要的安装包

```
yum install -y yum-utils
```

### 设置镜像仓库

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

```
yum-config-manager \--add-repo \http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo(阿里云)
```

## 安装相关 docker-ce社区有

---

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

## 启动 Docker

---

### systemctl start docker

```
systemctl start docker
```

查看 版本  
docker version

## 重启 Docker

---

### systemctl restart docker

```
systemctl restart docker
```

查看 版本  
docker version

## Hello-World

---

```
docker run hello-world
```

```
[root@hoyin ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b8dfdel27a29: Pull complete
Digest: sha256:9f6ad537c5132bcce57f7a0a20e317228d382c3cd61edae14650eec68b2b345c
Status: Downloaded newer image for hello-world:latest

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@hoyin ~]#
```

## 查看镜像

---

```
docker images
```

## 卸载docker

---

### yum remove docker-ce docker-ce-cli containerd.io

```
yum remove docker-ce docker-ce-cli containerd.io
```

删除 资源

```
rm -rf /var/lib/docker
```

```
rm -rf /var/lib/containerd
```

## 配置阿里云镜像加速

---

阿里云

容器镜像服务

镜像工具

容器镜像服务

实例列表

镜像中心

镜像工具

镜像加速器

容器镜像服务 / 镜像加速器

镜像加速器

使用加速器可以提升获取Docker官方镜像的速度

加速器

加速器地址

<https://2r8fac7y.mirror.aliyuncs.com> 复制

操作文档

Ubuntu CentOS Mac Windows

1. 安装 / 升级Docker客户端

推荐安装 1.10.0 以上版本的Docker客户端，参考文档[docker-ce](#)

2. 配置镜像加速器

针对Docker客户端版本大于 1.10.0 的用户

您可以通过修改daemon配置文件 `/etc/docker/daemon.json` 来使用加速器

```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<- 'EOF'
{
  "registry-mirrors": ["https://2r8fac7y.mirror.aliyuncs.com"]
}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

## 配置使用

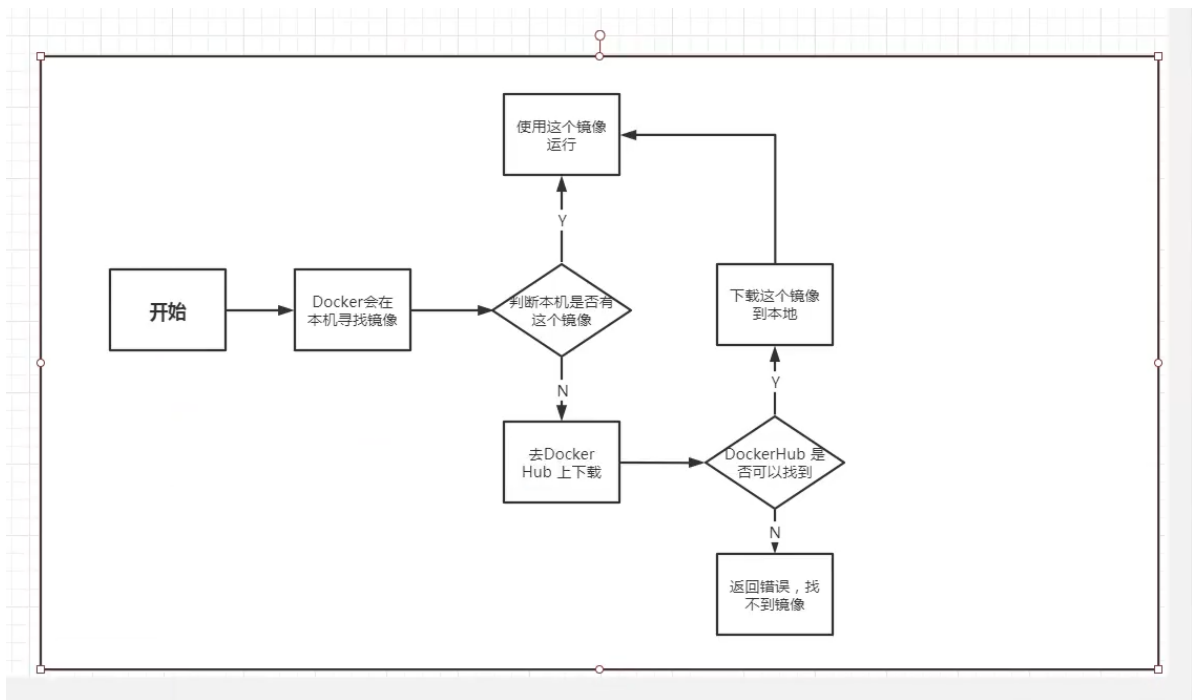
```
sudo mkdir -p /etc/docker

sudo tee /etc/docker/daemon.json <<- 'EOF'
{
  "registry-mirrors": ["https://2r8fac7y.mirror.aliyuncs.com"]
}
EOF

sudo systemctl daemon-reload

sudo systemctl restart docker
```

## docker run

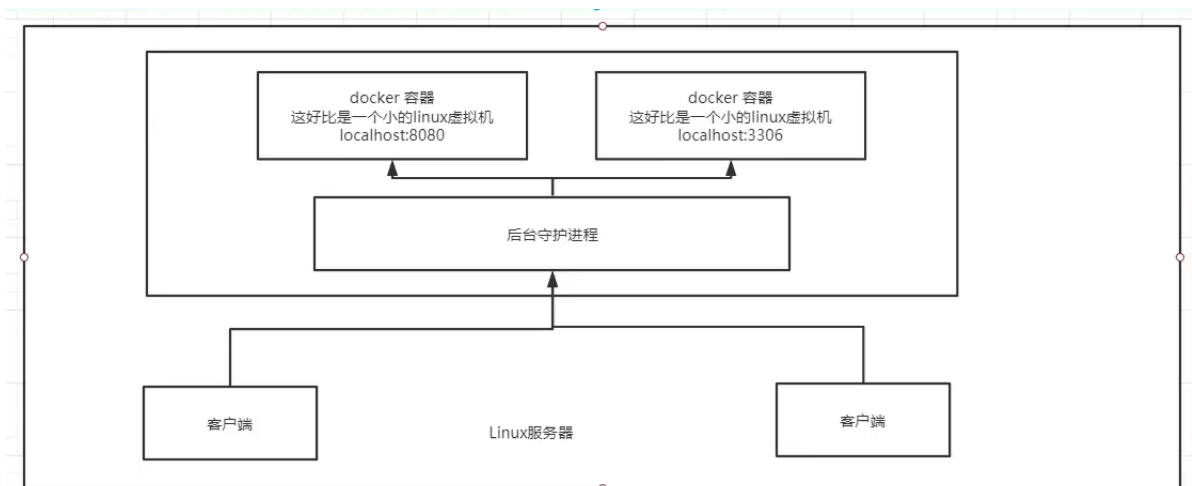


## 底层原理

### Docker 是怎么工作的？

Docker 是一个Client-Server 结构的系统，Docker的守护进程运行再主机上。通过Socket从客户端访问！

DockerServer 接收到Docker-Client的指令，就会执行这个命令！



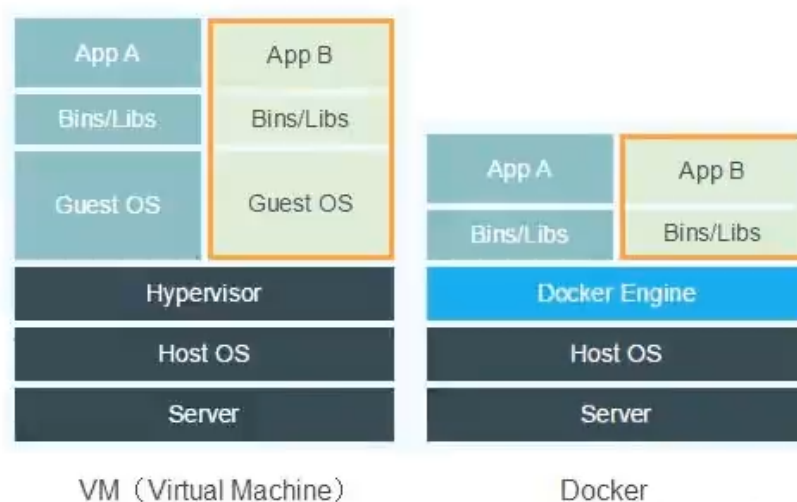
### Docker 为啥比VM快

- 1 .Docker有着比虚拟机更少的抽象层
- 2 .Docker利用的是宿主机的内核，vm需要的是Guest Os.

所以，新建一个容器的时候：

虚拟机是加载Guest Os,分钟级！

Docker不需要像虚拟机一样重新加载一个操作系统的内核，避免引导。而Docker利用的是宿主机的操作系统。省略了这个复杂的过程，秒级！



## Docker 常用命令

### 帮助命令

```
docker version    #显示版本信息
docker info       #显示系统信息
docker 命令 --help #帮助命令
```

帮助文档地址: <https://docs.docker.com/reference/>

### 镜像命令

#### docker images 镜像列表

查看所有本地的主机上的镜像

```
[root@hoyin ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    d1165f221234   3 months ago  13.3kB
```

```
#
REPOSITORY  镜像的仓库源
TAG          镜像标签
IMAGE ID     镜像id
CREATED      创建时间
SIZE         大小

# 可选项
-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 image IDs

docker images -aq
```

## docker search 镜像搜索

```
[root@hoyin ~]# docker search mysql
NAME                                DESCRIPTION
STARS      OFFICIAL    AUTOMATED
mysql                               MySQL is a widely used, open-source relation...
11023      [OK]
mariadb                               MariaDB Server is a high performing open sou...
4177       [OK]

# 可选项
```

## docker pull 下载镜像

下载镜像 docker pull 镜像名字[:tag]

### docker pull mysql

```
[root@hoyin ~]# docker pull mysql
Using default tag: latest # #不写默认最新 latest
latest: Pulling from library/mysql
69692152171a: Pull complete #分层下载
1651b0be3df3: Pull complete
951da7386bc8: Pull complete
0f86c95aa242: Pull complete
37ba2d8bd4fe: Pull complete
6d278bb05e94: Pull complete
497efbd93a3e: Pull complete
f7fddf10c2c2: Pull complete
16415d159dfb: Pull complete
0e530ffc6b73: Pull complete
b0a4a1a77178: Pull complete
cd90f92aa9ef: Pull complete
Digest: sha256:d50098d7fcb25b1fcb24e2d3247cae3fc55815d64fec640dc395840f8fa80969
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest #真实地址
```

#等价

```
docker pull mysql == docker pull docker.io/library/mysql:latest
```

## #tag docker pull mysql:5.7

```
[root@hoyin ~]# docker pull mysql:5.7
5.7: Pulling from library/mysql
69692152171a: Already exists
1651b0be3df3: Already exists
951da7386bc8: Already exists
0f86c95aa242: Already exists
37ba2d8bd4fe: Already exists
6d278bb05e94: Already exists
497efbd93a3e: Already exists
a023ae82eef5: Pull complete
e76c35f20ee7: Pull complete
e887524d2ef9: Pull complete
ccb65627e1c3: Pull complete
Digest: sha256:a682e3c78fc5bd941e9db080b4796c75f69a28a8cad65677c23f7a9f18ba21fa
Status: Downloaded newer image for mysql:5.7
docker.io/library/mysql:5.7
```

## docker rmi 删除镜像

```
docker rmi -f 容器Id           #删除指定的容器

docker rmi -f 容器Id 容器Id 容器Id  #删除多个容器

docker rmi -f $(docker images -aq)  #删除全部容器
```

## docker rmi -f imageld

```
[root@hoyin ~]# docker rmi -f 2c9028880e58
Untagged: mysql:5.7
Untagged:
mysql@sha256:a682e3c78fc5bd941e9db080b4796c75f69a28a8cad65677c23f7a9f18ba21fa
Deleted: sha256:2c9028880e5814e8923c278d7e2059f9066d56608a21cd3f83a01e3337bacd68
Deleted: sha256:c49c5c776f1bc87cdfff451ef39ce16a1ef45829e10203f4d9a153a6889ec15e
Deleted: sha256:8345316eca77700e62470611446529113579712a787d356e5c8656a41c244aee
Deleted: sha256:8ae51b87111404bd3e3bde4115ea2fe3fd2bb2cf67158460423c361a24df156b
Deleted: sha256:9d5afda6f6dcf8dd59aef5c02099f1d3b3b0c9ae4f2bb7a61627613e8cdf562
```

## docker rmi -f \$(docker images -aq)

```
[root@hoyin ~]# docker rmi -f $(docker images -aq)
Untagged: mysql:latest
Untagged:
mysql@sha256:d50098d7fcb25b1fcb24e2d3247cae3fc55815d64fec640dc395840f8fa80969
Deleted: sha256:c0cdc95609f1fc1daf2c7cae05ebd6adcf7b5c614b4f424949554a24012e3c09
Deleted: sha256:137bebc5ea278e61127e13cc7312fd83874cd19e392ee87252b532f0162fbd56
Deleted: sha256:7ed0de2ad4e43c97f58fa9e81fba73700167ef9f8a9970685059e0109749a56b
Deleted: sha256:9375660fbff871cd29c86b8643be60e376bfc96e99a3d7e8f93d74cd61500705
```



```
Deleted: sha256:d8a47065d005ac34d81017677330ce096eb5562eeb971e2db12b0e200fdb1cb6
Deleted: sha256:ca13c8ad9df5d824d5a259a927eaa6c04a60f022bc2abe8fc7866cf4b2b366f4
Deleted: sha256:7af1865d5c19316c3dc0829a2ee2b3a744ae756f7fec9c213d3afc5f1f6ed306
Deleted: sha256:f205c8f3c8aaa6376442b34c0c2062738461d37e0aa16ba021cd7e09c67213c2
Deleted: sha256:d690e8a8242cf13cbe98c5b2faffdd0fc7e6c4c13425b5da31de991aa1f89a76
Deleted: sha256:24efeee958e9f3d859fe15540e9296d5aaa6d3eb3b5f5494a2e8370608a4cfaa
Deleted: sha256:654f2ffede3bb536fd62d04c9c7b7826e890828bec92182634e38684959b2498
Deleted: sha256:de478a06eaa676052e665faa0b07d86a007f4b87cf82eb46a258742dc2d32260
Deleted: sha256:02c055ef67f5904019f43a41ea5f099996d8e7633749b6e606c400526b2c4b33
Untagged: hello-world:latest
Untagged: hello-
world@sha256:9f6ad537c5132bcce57f7a0a20e317228d382c3cd61edae14650eec68b2b345c
Deleted: sha256:d1165f2212346b2bab48cb01c1e39ee8ad1be46b87873d9ca7a4e434980a7726
```

---

## 容器命令

### 新建容器

```
docker pull centos
```

### 启动容器 docker run

```
docker run [可选参数] image
# 参数说明
--name="name"    容器名字
-d              后台运行
-it            使用交互方式运行，今日容器查看内容
-p(小写)       指定容器的端口 -p 8080:8080
    -p ip:主机端口:容器端口
    -p 主机端口:容器端口 （常用）
    -p 容器端口
    容器端口
-P(大写)       随机指定端口
-v            卷挂载
-e            环境配置
```

```
[root@hoyin ~]# docker run -it centos /bin/bash
[root@a44628968477 /]#
```

### docker ps 列出所有运行的容器

```
# docker ps 命令
-a          # 列出当前正在运行的容器 + 带出历史运行的容器
-n=?       # 显示最近创建的容器
-q         # 显示容器编号

[root@hoyin ~]# docker ps
```

| CONTAINER ID                        | IMAGE        | COMMAND           | CREATED       | STATUS       | PORTS         | NAMES |
|-------------------------------------|--------------|-------------------|---------------|--------------|---------------|-------|
| [root@hoyin ~]# <b>docker ps -a</b> |              |                   |               |              |               |       |
| CONTAINER ID                        | IMAGE        | COMMAND           | CREATED       | STATUS       | PORTS         | NAMES |
| 0fe4cd02b1ca                        | centos       | "/bin/bash"       | 3 minutes ago | Exited (127) | 39            |       |
| a44628968477                        | centos       | test              | 4 minutes ago | Exited (0)   | 4 minutes ago |       |
| df54ca90b778                        | centos       | objective_kapitsa | 5 minutes ago | Exited (130) | 5             |       |
| d14e89c50b1a                        | d1165f221234 | "/hello"          | 2 days ago    | Exited (0)   | 2 days ago    |       |
| 41831354804a                        | d1165f221234 | "/hello"          | 5 days ago    | Exited (0)   | 5 days ago    |       |
|                                     |              | youthful_gauss    |               |              |               |       |
|                                     |              | elegant_herschel  |               |              |               |       |
|                                     |              | pensive_cannon    |               |              |               |       |

## exit 直接容器停止并退出容器

```
exit # 直接容器停止并退出
Ctrl + P + Q # 容器不停止退出
[root@a44628968477 /]# exit
exit
```

## Ctrl + P + Q # 容器不停止退出

```
exit # 直接容器停止并退出
Ctrl + P + Q # 容器不停止退出
[root@a44628968477 /]# exit
exit
```

## docker rm 删除容器

```
docker rm 容器id #删除指定的容器，不能删除运行的容器

docker rm -f $(docker ps -aq) #删除所有容器

docker ps -a -q|xargs docker rm #删除所有容器
```

## docker start 启动容器

```
docker start 容器Id #启动容器
```

## docker restart 重启容器

```
docker restart 容器Id #重启容器
```

## docker stop 停止容器

```
docker stop 容器Id #停止当前正在运行的容器
```

## docker kill 强制停止

```
docker kill 容器Id #强制停止
```

## 总结

```
docker start 容器Id #启动容器
docker restart 容器Id #重启容器
docker stop 容器Id #停止当前正在运行的容器
docker kill 容器Id #强制停止
```

## 常用其他命令

### docker run -d 镜像名 [后台启动容器]

#### docker run -d 镜像名

```
# 命令 docker run -d 镜像名
[root@hoyin ~]# docker run -d centos
f5727dbc7d895113874bbd07674db77a8af99c019c1f4c2443cd514520b76030
[root@hoyin ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
#注意事项
docker 容器使用后台运行，就必须要有有一个前台进程，docker发现没有应用，就会自动停止。
容器启动后，发现自己没有提供服务，就会立即停止。就会没有程序
```

#### -e 添加环境配置

```
docker run -d --name elasticsearch02 -p 9200:9200 -p 9300:9300 -e
"discovery.type=single-node"-eES_JAVA_OPTS="-Xms64m -Xmx512m"
elasticsearch:7.6.2
```

### docker logs [查看logs]

#### docker logs -ft --tail number 容器Id

```
[root@hoyin ~]# docker logs -ft --tail 10 db526f11e349
# shell
[root@hoyin ~]# docker run -d centos /bin/sh -c "while true;do echo shoyin;sleep
1;done"
```

```
439d22a43c0e0415db7aa9d497ab078d27bc575172e5c9272771ccbeecf83e8c
```

```
# 显示日志
```

```
-tf
```

```
--tail number
```

```
[root@hoyin ~]# docker logs -tf --tail 10 439d22a43c0e
```

```
2021-06-21T08:12:11.907509823Z shoyin
```

```
2021-06-21T08:12:12.909276636Z shoyin
```

```
2021-06-21T08:12:13.911111547Z shoyin
```

```
2021-06-21T08:12:14.912899374Z shoyin
```

```
2021-06-21T08:12:15.914727760Z shoyin
```

```
2021-06-21T08:12:16.916464842Z shoyin
```

```
2021-06-21T08:12:17.918250190Z shoyin
```

```
2021-06-21T08:12:18.920114482Z shoyin
```

```
2021-06-21T08:12:19.921931290Z shoyin
```

## docker top [查看容器中的进程信息]

docker top 容器Id

```
[root@hoyin ~]# docker top e5c0e34be361
```

| UID                              | PID     | PPID     | C                    |
|----------------------------------|---------|----------|----------------------|
| STIME                            | TTY     | TIME     | CMD                  |
| root                             | 3984344 | 3984323  | 1                    |
| 16:16                            | ?       | 00:00:00 | /bin/sh -c while     |
| true;do echo shoyin;sleep 1;done |         |          |                      |
| root                             | 3985078 | 3984344  | 0                    |
| 16:17                            | ?       | 00:00:00 | /usr/bin/coreutils - |
| -coreutils-prog-shebang=sleep    |         |          |                      |
| /usr/bin/sleep 1                 |         |          |                      |

## docker inspect [查看镜像的元数据]

docker inspect 容器Id

```
[root@hoyin ~]# docker inspect e5c0e34be361
```

```
[
  {
    "Id":
    "e5c0e34be3611d3c2b0e121262372243a23b46c5b5e540760208bb747c5c26b0",
    "Created": "2021-06-21T08:16:40.617635229Z",
    "Path": "/bin/sh",
    "Args": [
      "-c",
      "while true;do echo shoyin;sleep 1;done"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 3984344,
      "ExitCode": 0,
```

```
        "Error": "",
        "StartedAt": "2021-06-21T08:16:41.223988268Z",
        "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image":
    "sha256:300e315adb2f96afe5f0b2780b87f28ae95231fe3bdd1e16b9ba606307728f55",
    "ResolvConfPath":
    "/var/lib/docker/containers/e5c0e34be3611d3c2b0e121262372243a23b46c5b5e540760208bb747c5c26b0/resolv.conf",
    "HostnamePath":
    "/var/lib/docker/containers/e5c0e34be3611d3c2b0e121262372243a23b46c5b5e540760208bb747c5c26b0/hostname",
    "HostsPath":
    "/var/lib/docker/containers/e5c0e34be3611d3c2b0e121262372243a23b46c5b5e540760208bb747c5c26b0/hosts",
    "LogPath":
    "/var/lib/docker/containers/e5c0e34be3611d3c2b0e121262372243a23b46c5b5e540760208bb747c5c26b0-
    json.log",
    "Name": "/laughing_cerf",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "",
    "ExecIDs": null,
    "HostConfig": {
        "Binds": null,
        "ContainerIDFile": "",
        "LogConfig": {
            "Type": "json-file",
            "Config": {}
        },
        "NetworkMode": "default",
        "PortBindings": {},
        "RestartPolicy": {
            "Name": "no",
            "MaximumRetryCount": 0
        },
        "AutoRemove": false,
        "VolumeDriver": "",
        "VolumesFrom": null,
        "CapAdd": null,
        "CapDrop": null,
        "CgroupnsMode": "host",
        "Dns": [],
        "DnsOptions": [],
        "DnsSearch": [],
        "ExtraHosts": null,
        "GroupAdd": null,
        "IpcMode": "private",
        "Cgroup": "",
        "Links": null,
        "OomScoreAdj": 0,
        "PidMode": "",
        "Privileged": false,
        "PublishAllPorts": false,
```

```
"ReadonlyRootfs": false,
"SecurityOpt": null,
"UTSMode": "",
"UsersnsMode": "",
"ShmSize": 67108864,
"Runtime": "runc",
"ConsoleSize": [
    0,
    0
],
"Isolation": "",
"CpuShares": 0,
"Memory": 0,
"NanoCpus": 0,
"CgroupParent": "",
"Blkioweight": 0,
"BlkioweightDevice": [],
"BlkioDeviceReadBps": null,
"BlkioDeviceWriteBps": null,
"BlkioDeviceReadIops": null,
"BlkioDeviceWriteIops": null,
"CpuPeriod": 0,
"CpuQuota": 0,
"CpuRealtimePeriod": 0,
"CpuRealtimeRuntime": 0,
"CpusetCpus": "",
"CpusetMems": "",
"Devices": [],
"DeviceCgroupRules": null,
"DeviceRequests": null,
"KernelMemory": 0,
"KernelMemoryTCP": 0,
"MemoryReservation": 0,
"MemorySwap": 0,
"MemorySwappiness": null,
"OomKillDisable": false,
"PidsLimit": null,
"Ulimits": null,
"CpuCount": 0,
"CpuPercent": 0,
"IOMaximumIops": 0,
"IOMaximumBandwidth": 0,
"MaskedPaths": [
    "/proc/asound",
    "/proc/acpi",
    "/proc/kcore",
    "/proc/keys",
    "/proc/latency_stats",
    "/proc/timer_list",
    "/proc/timer_stats",
    "/proc/sched_debug",
    "/proc/scsi",
    "/sys/firmware"
],
"ReadonlyPaths": [
    "/proc/bus",
    "/proc/fs",
    "/proc/irq",
```

```

        "/proc/sys",
        "/proc/sysrq-trigger"
    ]
},
"GraphDriver": {
    "Data": {
        "LowerDir":
"/var/lib/docker/overlay2/9b7b52784a0aaddcd63a43094d571b13d4f6f34ff842f9141aba6d1e516a10c8-
init/diff:/var/lib/docker/overlay2/9d30ff772de9466d3dc90840317269af610265c3a04bc
ebdcd5c3b365818e5e/diff",
        "MergedDir":
"/var/lib/docker/overlay2/9b7b52784a0aaddcd63a43094d571b13d4f6f34ff842f9141aba6d1e516a10c8/merged",
        "UpperDir":
"/var/lib/docker/overlay2/9b7b52784a0aaddcd63a43094d571b13d4f6f34ff842f9141aba6d1e516a10c8/diff",
        "WorkDir":
"/var/lib/docker/overlay2/9b7b52784a0aaddcd63a43094d571b13d4f6f34ff842f9141aba6d1e516a10c8/work"
    },
    "Name": "overlay2"
},
"Mounts": [],
"Config": {
    "Hostname": "e5c0e34be361",
    "Domainname": "",
    "User": "",
    "AttachStdin": false,
    "AttachStdout": false,
    "AttachStderr": false,
    "Tty": false,
    "OpenStdin": false,
    "StdinOnce": false,
    "Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
    ],
    "Cmd": [
        "/bin/sh",
        "-c",
        "while true;do echo shoyin;sleep 1;done"
    ],
    "Image": "centos",
    "Volumes": null,
    "WorkingDir": "",
    "Entrypoint": null,
    "OnBuild": null,
    "Labels": {
        "org.label-schema.build-date": "20201204",
        "org.label-schema.license": "GPLv2",
        "org.label-schema.name": "CentOS Base Image",
        "org.label-schema.schema-version": "1.0",
        "org.label-schema.vendor": "CentOS"
    }
},
"NetworkSettings": {
    "Bridge": "",

```

```

        "SandboxID":
        "b512756b2e28fbd198be5cdc02192cc92a43bd0c06046480c4db4c3e82ef6021",
        "HairpinMode": false,
        "LinkLocalIPv6Address": "",
        "LinkLocalIPv6PrefixLen": 0,
        "Ports": {},
        "SandboxKey": "/var/run/docker/netns/b512756b2e28",
        "SecondaryIPAddresses": null,
        "SecondaryIPv6Addresses": null,
        "EndpointID":
        "e8539fb44a08424cf7e2b41f94a4204d042de0ecb0b11a91ab656283e2be9312",
        "Gateway": "172.17.0.1",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "IPAddress": "172.17.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "MacAddress": "02:42:ac:11:00:02",
        "Networks": {
            "bridge": {
                "IPAMConfig": null,
                "Links": null,
                "Aliases": null,
                "NetworkID":
                "c9dbce87ad95c14f4154df0237dc6db907f9f87dfb7547b8a852c7ea8402a796",
                "EndpointID":
                "e8539fb44a08424cf7e2b41f94a4204d042de0ecb0b11a91ab656283e2be9312",
                "Gateway": "172.17.0.1",
                "IPAddress": "172.17.0.2",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "MacAddress": "02:42:ac:11:00:02",
                "DriverOpts": null
            }
        }
    }
}
]

```

## exec attach [进入当前正在运行的容器]

**docker exec -it 容器id bashShell**

```

[root@hoyin ~]# docker exec -it d71b5024d0cd /bin/bash
[root@d71b5024d0cd /]# exit
exit 退出后不停止容器

```



## docker attach 容器Id

```
[root@hoyin ~]# docker attach d71b5024d0cd
[root@d71b5024d0cd /]# exit
exit 退出后直接停止容器
```

#docker exec 开启新得终端 exit 退出后不停止容器  
#docker attach 正在运行的终端 exit 退出后直接停止容器

## docker cp [从容器内拷贝文件到主机]

docker cp 容器id: 容器内路径 主机路径

```
[root@hoyin home]# docker cp 942c1d016e5a:/home/test.js /home
[root@hoyin home]# ls
test.js
```

docker cp 容器Name: 容器内路径 主机路径

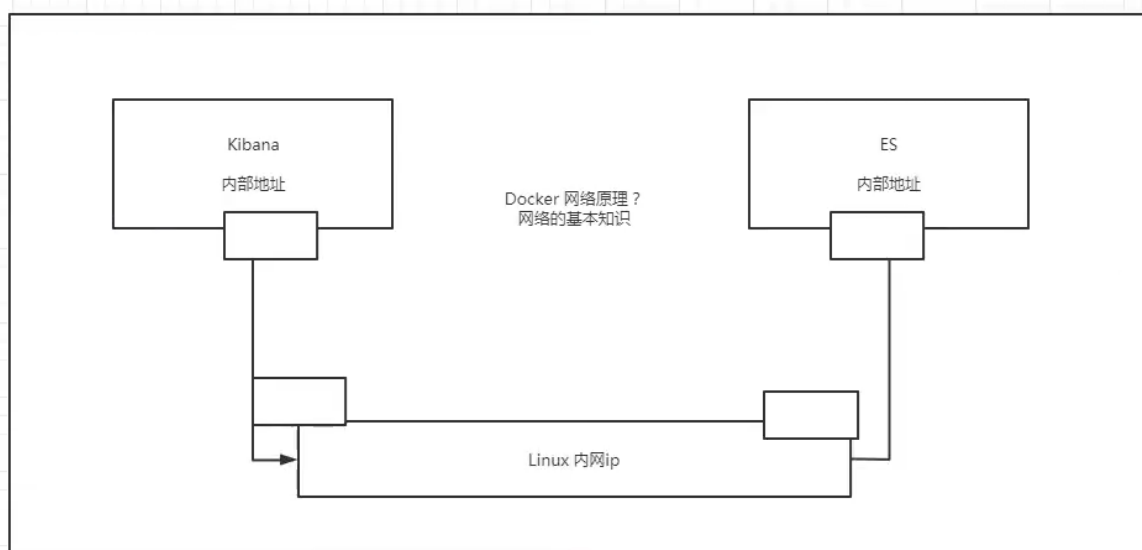
```
[root@hoyin home]# docker cp centos09:/home/test.js /home
[root@hoyin home]# ls
test.js
```

## docker stats

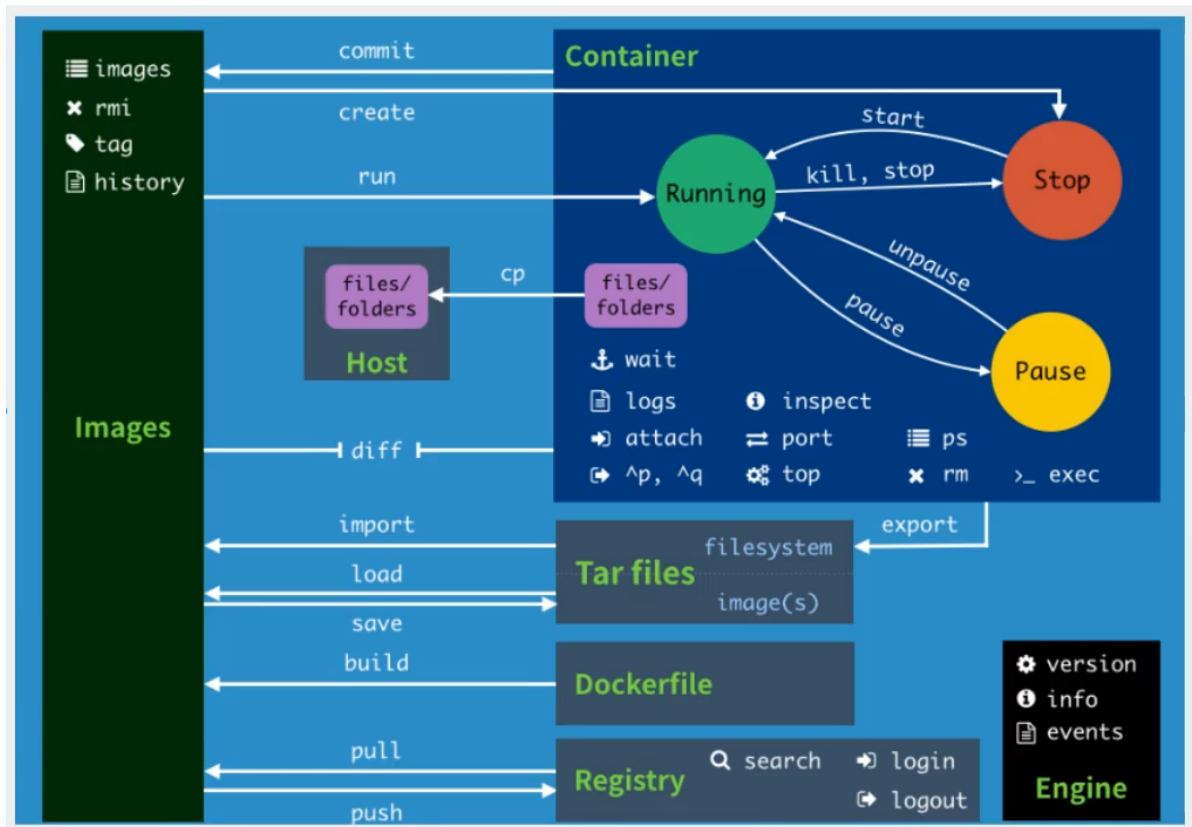
| CONTAINER ID | NAME     | CPU % | MEM USAGE / LIMIT   | MEM % | NET I/O     |
|--------------|----------|-------|---------------------|-------|-------------|
| 4d7d83c742a1 | centos09 | 0.00% | 1.273MiB / 1.776GiB | 0.07% | 1.41kB / 0B |
| 99dccf3551f9 | nginx01  | 0.00% | 2.309MiB / 1.776GiB | 0.13% | 17.1kB /    |
|              |          |       |                     |       |             |

| BLOCK I/O              | PIDS |
|------------------------|------|
| 4.1kB / 0B             | 1    |
| 17.7kB 8.19kB / 17.9kB | 2    |



## 小结



## Docker 安装 Nginx

`docker search nginx`

`docker pull nginx`

`docker run -d --name nginx01 -p 3344:80 nginx`

```
[root@hoyin ~]# docker search nginx
```

| NAME                        | STARS | OFFICIAL | AUTOMATED | DESCRIPTION                                     |
|-----------------------------|-------|----------|-----------|---|
| nginx                       | 15034 | [OK]     |           | official build of Nginx.                        |
| jwilder/nginx-proxy         | 2038  |          | [OK]      | Automated Nginx reverse proxy for docker con... |
| richarvey/nginx-php-fpm     | 814   |          | [OK]      | Container running Nginx + PHP-FPM capable of... |
| jc21/nginx-proxy-manager    | 199   |          |           | Docker container for managing Nginx proxy ho... |
| linuxserver/nginx           | 147   |          |           | An Nginx container, brought to you by LinuxS... |
| tiangolo/nginx-rtmp         | 130   |          | [OK]      | Docker image with Nginx using the nginx-rtmp... |
| jlesage/nginx-proxy-manager | 118   |          | [OK]      | Docker container for Nginx Proxy Manager        |

|                                  |      |   |
|----------------------------------|------|---|
| alfg/nginx-rtmp                  |      | NGINX, nginx-rtmp-module and FFmpeg from sou... |
| 99                               | [OK] |   |
| bitnami/nginx                    |      | Bitnami nginx Docker Image                      |
| 98                               | [OK] |   |
| nginxdemos/hello                 |      | NGINX webserver that serves a simple page co... |
| 70                               | [OK] |   |
| nginx/nginx-ingress              |      | NGINX and NGINX Plus Ingress Controllers fo...  |
| 55                               |      |   |
| privatebin/nginx-fpm-alpine      |      | PrivateBin running on an Nginx, php-fpm & Al... |
| 55                               | [OK] |   |
| nginxinc/nginx-unprivileged      |      | Unprivileged NGINX Dockerfiles                  |
| 38                               |      |   |
| staticfloat/nginx-certbot        |      | Opinionated setup for automatic TLS certs lo... |
| 23                               | [OK] |   |
| schmunk42/nginx-redirect         |      | A very simple container to redirect HTTP tra... |
| 19                               | [OK] |   |
| nginx/nginx-prometheus-exporter  |      | NGINX Prometheus Exporter for NGINX and NGIN... |
| 18                               |      |   |
| centos/nginx-112-centos7         |      | Platform for running nginx 1.12 or building ... |
| 15                               |      |   |
| centos/nginx-18-centos7          |      | Platform for running nginx 1.8 or building n... |
| 13                               |      |   |
| bitwarden/nginx                  |      | The Bitwarden nginx web server acting as a r... |
| 11                               |      |   |
| flashspys/nginx-static           |      | Super Lightweight Nginx Image                   |
| 10                               | [OK] |   |
| bitnami/nginx-ingress-controller |      | Bitnami Docker Image for NGINX Ingress Contr... |
| 9                                | [OK] |   |
| mailu/nginx                      |      | Mailu nginx frontend                            |
| 8                                | [OK] |   |
| devilbox/nginx-stable            |      | Devilbox's Nginx stable (based on official N... |
| 4                                |      |   |
| ansibleplaybookbundle/nginx-apb  |      | An APB to deploy NGINX                          |
| 2                                | [OK] |   |
| wodby/nginx                      |      | Generic nginx                                   |
| 1                                | [OK] |   |

```

[root@hoyin ~]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
69692152171a: Pull complete
30afc0b18f67: Pull complete
596b1d696923: Pull complete
febe5bd23e98: Pull complete
8283eee92e2f: Pull complete
351ad75a6cfa: Pull complete
Digest: sha256:6d75c99af15565a301e48297fa2d121e15d80ad526f8369c526324f0f7ccb750
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
[root@hoyin ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx         latest    d1a364dc548d   3 weeks ago    133MB
centos        latest    300e315adb2f   6 months ago   209MB
[root@hoyin ~]# docker run -d --name nginx01 -p 3344:80 nginx
d6d909ad6d7feab037772c2e289bc1ed48283c7762084990e903118568ef0f79
[root@hoyin ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
PORTS
NAMES

```

|                                       |               |                         |                |               |
|---------------------------------------|---------------|-------------------------|----------------|---------------|
| d6d909ad6d7f                          | nginx         | "/docker-entrypoint..." | 6 seconds ago  | Up 5 seconds  |
| 0.0.0.0:8080->80/tcp, :::8080->80/tcp | nginx01       |                         |                |               |
| 942c1d016e5a                          | centos        | "/bin/bash"             | 58 minutes ago | Up 55 minutes |
|                                       | hungry_keldys |                         |                |               |

## 本机测试

curl ip地址:3344

curl localhost:3344

```
[root@hoyin ~]# curl ip:3344
<!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>
```

---

## 可视化

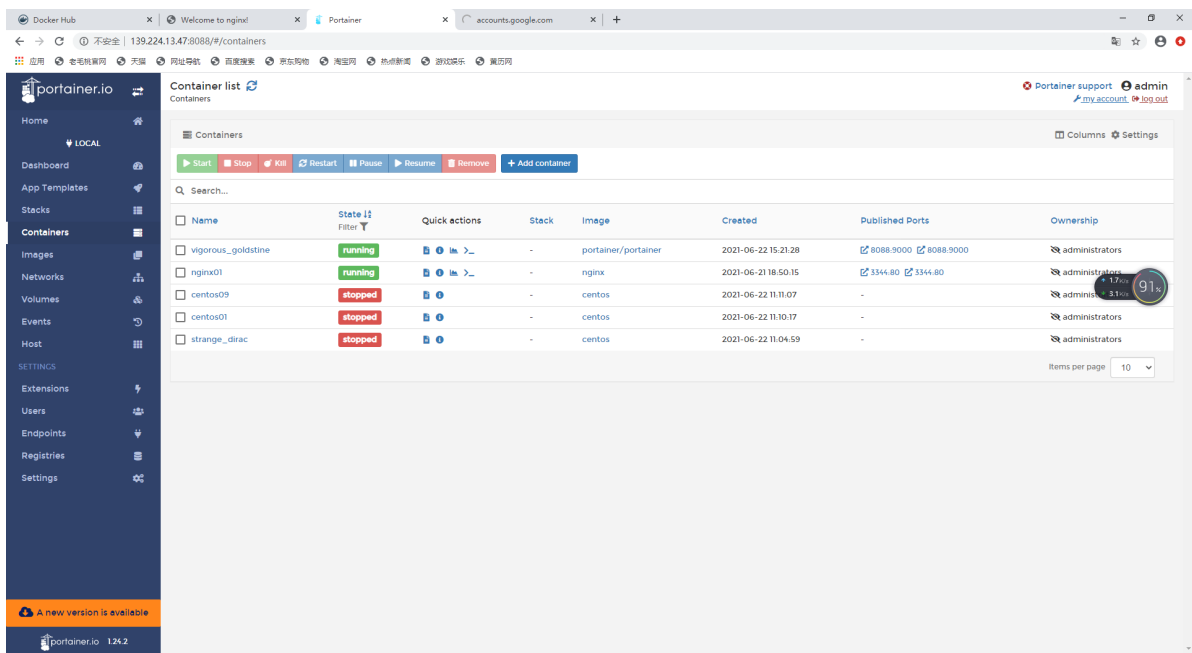
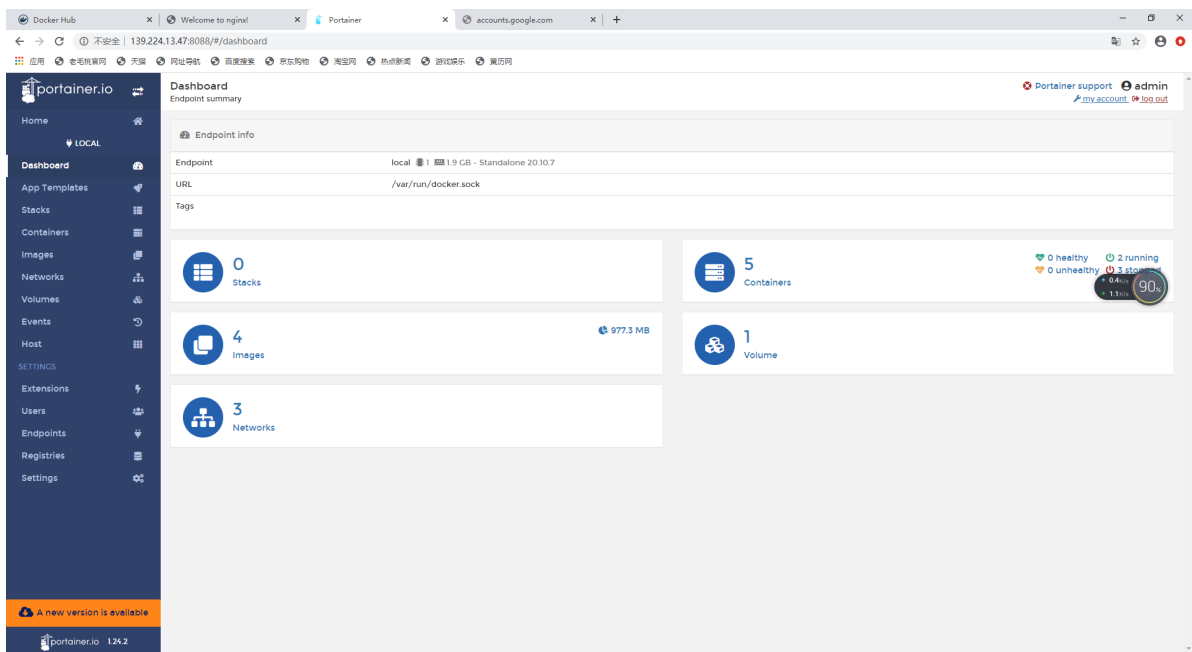
### portainer

Docker 图像可视化面板

```
[root@hoyin ~]# docker run -d -p 8088:9000 --restart=always -v
/var/run/docker.sock:/var/run/docker.sock --privileged=true portainer/portainer
Unable to find image 'portainer/portainer:latest' locally
latest: Pulling from portainer/portainer
94cfa856b2b1: Pull complete
49d59ee0881a: Pull complete
a2300fd28637: Pull complete
Digest: sha256:fb45b43738646048a0a0cc74fcee2865b69efde857e710126084ee5de9be0f3f
Status: Downloaded newer image for portainer/portainer:latest
57706419af92470fbdd5722c2f80ed2422ef6cedd693f12f8575d695bf50c822
```

## 访问测试

<http://ip:8088>



# Docker 镜像

## commit 镜像

`docker commit -m "提交信息centos10" -a "作者hoyin" 源容器id newName:[tag]`

`docker commit -m "提交信息centos10" -a "作者hoyin" 源容器Name newName:[tag]`

#1 启动一个默认的centos

#2 修改部分文件

#3 将我们操作过的容器通过commit提交为一个镜像!我们以后就使用我们修改过的镜像即可，这就是我们自己的一个修改的镜像

```
[root@hoyin ~]# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED
STATUS        PORTS              NAMES
57706419af92   portainer/portainer  "/portainer"           38 minutes ago
Up 38 minutes  0.0.0.0:8088->9000/tcp, :::8088->9000/tcp  vigorous_goldstine
4d7d83c742a1   centos               "/bin/bash"            5 hours ago
Up 3 minutes   centos09
99dccf3551f9   nginx               "/docker-entrypoint...." 21 hours ago
Up 46 minutes  0.0.0.0:3344->80/tcp, :::3344->80/tcp      nginx01

# commit id
[root@hoyin ~]# docker commit -m "centos10" -a "hoyin" 4d7d83c742a1 centos10:1.0
sha256:3058c2cc4c41b8c07bc064c758cb7263ad661d2d7d52c8809893d2c6ccb53651
[root@hoyin ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
centos10      1.0       3058c2cc4c41  7 seconds ago  210MB
nginx         latest    d1a364dc548d  3 weeks ago   133MB
mysql         latest    c0cdc95609f1  5 weeks ago   556MB
portainer/portainer latest    580c0e4e98b0  3 months ago  79.1MB
centos        latest    300e315adb2f  6 months ago  209MB

#commit name
#提交的镜像必须是正在运行中的镜像
[root@hoyin ~]# docker commit -m "centos11" -a "hoyin" centos09 centos11:1.0
sha256:16923a9b9678fce09b5cbc85b124d199e54496d872025ed42f225c07552302c5
[root@hoyin ~]# docker commit -m "centos11" -a "hoyin" centos10 centos11:1.0
Error response from daemon: No such container: centos10
```

# Docker 数据卷

容器的持久化和同步操作

容器间数据共享

## 使用数据卷

## docker run -it -v 主机目录:容器目录 容器Name bashShell

```
docker run -it -v /home/test:/home centos10 /bin/bash
```

```
docker inspect 容器id
```

```
    },
    "Name": "overlay2"
  },
  "Mounts": [
    {
      "Type": "bind",
      "Source": "/home/test",
      "Destination": "/home",
      "Mode": "",
      "RW": true,
      "Propagation": "rprivate"
    }
  ],
  "Config": {
    "Hostname": "3ac9d2611657",
```

## 安装Mysql

```
[root@hoyin test]# docker pull mysql:5.7
5.7: Pulling from library/mysql
69692152171a: Already exists
1651b0be3df3: Already exists
951da7386bc8: Already exists
0f86c95aa242: Already exists
37ba2d8bd4fe: Already exists
6d278bb05e94: Already exists
497efbd93a3e: Already exists
a023ae82eef5: Pull complete
e76c35f20ee7: Pull complete
e887524d2ef9: Pull complete
ccb65627e1c3: Pull complete
Digest: sha256:a682e3c78fc5bd941e9db080b4796c75f69a28a8cad65677c23f7a9f18ba21fa
Status: Downloaded newer image for mysql:5.7
docker.io/library/mysql:5.7
```

## # mysql 务必设置初始密码

# mysql 务必设置初始密码

```
docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag
```

```
docker run -d -p 3310:3306 -v /home/mysql/conf:/etc/mysql/conf.d -v  
/home/mysql/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=123456 --name mysql01  
mysql:5.7
```

```
[root@hoyin test]# docker run -d -p 3310:3306 -v  
/home/mysql/conf:/etc/mysql/conf.d -v /home/mysql/data:/var/lib/mysql -e  
MYSQL_ROOT_PASSWORD=123456 --name mysql01 mysql:5.7  
2f011b57949081bd0d8e00d2db2222aa2992800e9ac7020d46edf49a2a2f767a
```

## 具名和匿名挂载

### docker volume [查看挂载信息]

```
[root@hoyin test]# docker volume ls  
DRIVER      VOLUME NAME  
local       5ca43c35e2ffa9a6018a043cc14998b806fe39e8c93fb8625da467339aa4c61c  
local       d3a6ea05aff012486e272486b2ddd7f2f08ea10801dbbd3e8e5a42734c9dfa7c
```

|                 |         |
|-----------------|---------|
| -v 容器内路径        | #匿名挂载   |
| -v 卷名: 容器路径     | #具名挂载   |
| -v /宿主机路径: 容器路径 | #指定路径挂载 |

## 匿名挂载

```
#-v 容器内路径  
docker run -d -P --name nginx -v /etc/nginx nginx
```

```
[root@hoyin test]# docker run -d -P --name nginx01 -v /ect/nginx nginx  
b5860ebee5811ed22b27b78c2cab7a3cd253a3a9ecc3e88b973de83f1700e934  
[root@hoyin test]# docker volume ls  
DRIVER      VOLUME NAME  
local       5ca43c35e2ffa9a6018a043cc14998b806fe39e8c93fb8625da467339aa4c61c  
local       d3a6ea05aff012486e272486b2ddd7f2f08ea10801dbbd3e8e5a42734c9dfa7c
```



## 具名挂载

#-v 卷名: 容器路径

```
docker run -d -P --name nginx02 -v juming-nginx:/ect/nginx nginx
```

```
[root@hoyin test]# docker run -d -P --name nginx02 -v ju-nginx:/ect/nginx nginx
c20e6bef6741ba1f577a0c6357f233524629f4ea78bcc6c6b539b86818622efe
[root@hoyin test]# docker volume ls
DRIVER      VOLUME NAME
local       5ca43c35e2ffa9a6018a043cc14998b806fe39e8c93fb8625da467339aa4c61c
local       d3a6ea05aff012486e272486b2ddd7f2f08ea10801dbbd3e8e5a42734c9dfa7c
local       ju-nginx #新增卷
[root@hoyin test]# docker volume inspect ju-nginx
[
  {
    "CreatedAt": "2021-06-22T17:12:38+08:00",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/ju-nginx/_data",
    "Name": "ju-nginx",
    "Options": null,
    "Scope": "local"
  }
]
```

## docker volume目录

所有的docker容器内的卷，没有指定情况下都是在

**/var/lib/docker/volumes/xxxx/\_data**

我们通过具名挂载可以方便的找到卷

## -v :ro :rw 改变读写权限

ro     readonly

rw     readwrite

```
[root@hoyin test]# docker run -d -P --name nginx02 -v ju-nginx:/ect/nginx:ro
nginx
[root@hoyin test]# docker run -d -P --name nginx02 -v ju-nginx:/ect/nginx:rw
nginx
```

#ro 只能通过宿主机操作，容器内没有权限，无法操作

## docker volume prune

Remove all unused local volumes

## Docker 已有容器挂载新目录

1. 关闭正在运行Docker容器
2. docker stop container-Id
3. 关闭Docker.service
4. systemctl stop docker
5. 修改容器config

6. 

```
vim /var/lib/docker/containers/container-ID/config.v2.json
#修改 MountPoints
"MountPoints":{"/ect/nginx":
{"Source":"/home/nginx","Destination":"/ect/nginx","RW":true,"Name":"","Driver":"","Type":"bind","Propagation":"rprivate","Spec":
{"Type":"bind","Source":"/home/nginx","Target":"/ect/nginx"},"SkipMountpointCreation":false}}

"MountPoints":{"/容器挂载目录":{"Source":"/宿主机挂载目录","Destination":"/容器挂载目录","RW":true,"Name":"","Driver":"","Type":"bind","Propagation":"rprivate","Spec":{"Type":"bind","Source":"/宿主机挂载目录","Target":"/容器挂载目录"},"SkipMountpointCreation":false}}
```

7. 保存重启 docker.service
8. systemctl start docker
9. 重启容器
10. docker start container-Id

---

## Dockerfile

### Dockerfile 介绍

dockerfile 是用来构建Docker镜像的文件!

构建步骤:

1. 编辑一个dockerfile 文件
2. docker build 构建成新镜像
3. docker run 运行镜像
4. docker push 发布 (DockerHub, 阿里云镜像仓库)

### Dockerfile 构建过程

基础知识:

### Dockerfile 指令

---

# Docker image 发布

## docker login

```
[root@hoyin docker-test-volumes]# docker login -u shoyin
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
```

## docker tag

```
[root@hoyin docker-test-volumes]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
test02        latest    10d7306f365e   4 hours ago    209MB
test01        latest    dbfdd81c9772   4 hours ago    209MB
centos01      latest    2b810fd8dd82   23 hours ago    209MB
nginx         latest    4f380adfc10f   25 hours ago    133MB
mysql         5.7       2c9028880e58   6 weeks ago    447MB
centos        latest    300e315adb2f   6 months ago    209MB
[root@hoyin docker-test-volumes]# docker tag dbfdd81c9772 shoyin/test01:0.1
[root@hoyin docker-test-volumes]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
test02        latest    10d7306f365e   4 hours ago    209MB
shoyin/test01  0.1       dbfdd81c9772   4 hours ago    209MB
test01        latest    dbfdd81c9772   4 hours ago    209MB
centos01      latest    2b810fd8dd82   23 hours ago    209MB
nginx         latest    4f380adfc10f   25 hours ago    133MB
mysql         5.7       2c9028880e58   6 weeks ago    447MB
centos        latest    300e315adb2f   6 months ago    209MB
```

## docker push

```
[root@hoyin docker-test-volumes]# docker push shoyin/test01:0.1
The push refers to repository [docker.io/shoyin/test01]
2653d992f4ef: Mounted from library/centos
0.1: digest:
sha256:ea9d30fdad33e84a6a90355483d0f325b38125facd34f93d3802afe0a828fb36 size:
529
```

## docker pull

```
[root@hoyin docker-test-volumes]# docker pull shoyin/test01:0.1
0.1: Pulling from shoyin/test01
7a0437f04f83: Pull complete
Digest: sha256:ea9d30fdad33e84a6a90355483d0f325b38125facd34f93d3802afe0a828fb36
Status: Downloaded newer image for shoyin/test01:0.1
docker.io/shoyin/test01:0.1
```

## dicker save 打包

```
$ docker save busybox > busybox.tar

$ ls -sh busybox.tar

2.7M busybox.tar

$ docker save --output busybox.tar busybox

$ ls -sh busybox.tar

2.7M busybox.tar

$ docker save -o fedora-all.tar fedora

$ docker save -o fedora-latest.tar fedora:latest

[root@hoyin docker-test-volumes]# docker save -o test.tar shoyin/test01:0.1
[root@hoyin docker-test-volumes]# ll
总用量 211468
-rw-r--r-- 1 root root      88 6月 24 12:00 dockerfile
-rw----- 1 root root 216536064 6月 24 16:50 test.tar
```

## docker load

**docker load** : 导入使用 [docker save](#) 命令导出的镜像。

### 语法

```
docker load [OPTIONS]
```

OPTIONS 说明:

- **--input, -i**: 指定导入的文件, 代替 STDIN。
- **--quiet, -q**: 精简输出信息。

### 实例

导入镜像:

```
$ docker image ls

REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE

$ docker load < busybox.tar.gz

Loaded image: busybox:latest
$ docker images
```

| REPOSITORY | TAG    | IMAGE ID     | CREATED     |
|------------|--------|--------------|-------------|
| SIZE       |        |              |             |
| busybox    | latest | 769b9341d937 | 7 weeks ago |
| 2.489 MB   |        |              |             |

```
$ docker load --input fedora.tar
```

```
Loaded image: fedora:rawhide
```

```
Loaded image: fedora:20
```

```
$ docker images
```

| REPOSITORY | TAG       | IMAGE ID     | CREATED     |
|------------|-----------|--------------|-------------|
| SIZE       |           |              |             |
| busybox    | latest    | 769b9341d937 | 7 weeks ago |
| 2.489 MB   |           |              |             |
| fedora     | rawhide   | 0d20aec6529d | 7 weeks ago |
| 387 MB     |           |              |             |
| fedora     | 20        | 58394af37342 | 7 weeks ago |
| 385.5 MB   |           |              |             |
| fedora     | heisenbug | 58394af37342 | 7 weeks ago |
| 385.5 MB   |           |              |             |
| fedora     | latest    | 58394af37342 | 7 weeks ago |
| 385.5 MB   |           |              |             |

```
[root@hoyin docker-test-volumes]# docker images
```

| REPOSITORY    | TAG | IMAGE ID     | CREATED     | SIZE  |
|---------------|-----|--------------|-------------|-------|
| shoyin/test01 | 0.1 | dbfdd81c9772 | 5 hours ago | 209MB |

```
[root@hoyin docker-test-volumes]# docker rmi -f shoyin/test01:0.1
```

```
Untagged: shoyin/test01:0.1
```

```
Untagged:
```

```
shoyin/test01@sha256:ea9d30fdad33e84a6a90355483d0f325b38125facd34f93d3802afe0a828fb36
```

```
Deleted: sha256:dbfdd81c97722101bc1f0ea0b243001a4bdf9ed1a1458620a698f0882a82274a
```

```
Deleted: sha256:2653d992f4ef2bfd27f94db643815aa567240c37732cae1405ad1c1309ee9859
```

```
[root@hoyin docker-test-volumes]# docker load -i test.tar
```

```
2653d992f4ef: Loading layer
```

```
[=====>] 216.5MB/216.5MB
```

```
Loaded image: shoyin/test01:0.1
```

```
[root@hoyin docker-test-volumes]# docker images
```

| REPOSITORY    | TAG | IMAGE ID     | CREATED     | SIZE  |
|---------------|-----|--------------|-------------|-------|
| shoyin/test01 | 0.1 | dbfdd81c9772 | 5 hours ago | 209MB |

```
[root@hoyin docker-test-volumes]#
```

## docker build

### 编辑Dockerfile

```
[root@hoyin docker-test-volumes]# vim dockerfile
```

### #命令行大写

```
FROM centos
VOLUME ["/volume01", "/volume02"]
CMD echo "-----end-----"
CMD /bin/bash
```

**VOLUME ["/volume01", "/volume02", "/绝对路径"]**

## #docker build

```
[root@hoyin docker-test-volumes]# docker build -f /home/docker-test-
volumes/dockerfile -t newcentos:1.0 .
#(. 不能忽略)
Sending build context to Docker daemon 2.048kB
Step 1/4 : FROM centos
--> 300e315adb2f
Step 2/4 : VOLUME ["/volume01", "/volume02"]
--> Running in dbb832595496
Removing intermediate container dbb832595496
--> c8ae5a49cbcf
Step 3/4 : CMD echo "-----end-----"
--> Running in d1fa81c53091
Removing intermediate container d1fa81c53091
--> 90cc79af129e
Step 4/4 : CMD /bin/bash
--> Running in e8d3f76991f7
Removing intermediate container e8d3f76991f7
--> 7de44945c539
Successfully built 7de44945c539
Successfully tagged newcentos:1.0
[root@hoyin docker-test-volumes]# docker images
```

| REPOSITORY          | TAG    | IMAGE ID     | CREATED        | SIZE   |
|---------------------|--------|--------------|----------------|--------|
| newcentos           | 1.0    | 7de44945c539 | 58 seconds ago | 209MB  |
| centos11            | 1.0    | 16923a9b9678 | 2 hours ago    | 210MB  |
| centos10            | 1.0    | 3058c2cc4c41 | 2 hours ago    | 210MB  |
| nginx               | latest | d1a364dc548d | 3 weeks ago    | 133MB  |
| mysql               | 5.7    | 2c9028880e58 | 5 weeks ago    | 447MB  |
| mysql               | latest | c0cdc95609f1 | 5 weeks ago    | 556MB  |
| portainer/portainer | latest | 580c0e4e98b0 | 3 months ago   | 79.1MB |
| centos              | latest | 300e315adb2f | 6 months ago   | 209MB  |

# 数据卷容器

## --volumes-from

```
# Build an image from a Dockerfile 根据Dockerfile 创建镜像
[root@hoyin docker-test-volumes]# docker build -f /home/docker-test-
volumes/dockerfile -t centos01 .
Sending build context to Docker daemon 2.048kB
Step 1/4 : FROM centos
latest: Pulling from library/centos
```

```

7a0437f04f83: Already exists
Digest: sha256:5528e8b1b1719d34604c87e11dcd1c0a20bedf46e83b5632cdeac91b8c04efc1
Status: Downloaded newer image for centos:latest
---> 300e315adb2f
Step 2/4 : VOLUME ["/test01"]
---> Running in bfab3286eb00
Removing intermediate container bfab3286eb00
---> 22bfc79835e7
Step 3/4 : CMD echo "-----end-----"
---> Running in 9e3ebb9ae31f
Removing intermediate container 9e3ebb9ae31f
---> 51edec24b776
Step 4/4 : CMD /bin/bash
---> Running in 47ba196bab02
Removing intermediate container 47ba196bab02
---> 2b810fd8dd82
Successfully built 2b810fd8dd82
Successfully tagged centos01:latest
[root@hoyin docker-test-volumes]# docker images
REPOSITORY    TAG       IMAGE ID      CREATED        SIZE
centos01      latest    2b810fd8dd82  32 seconds ago 209MB
nginx         latest    4f380adfc10f  3 hours ago    133MB
mysql         5.7       2c9028880e58  6 weeks ago    447MB
centos        latest    300e315adb2f  6 months ago   209MB
# 查看新生成镜像 创建新容器
[root@hoyin docker-test-volumes]# docker run -it --name docker01 2b810fd8dd82
[root@29e7111c640f /]# exit C+P+Q
# --volumes-from 继承容器挂载 创建新容器
[root@hoyin docker-test-volumes]# docker run -it --name docker02 --volumes-from
docker01 centos01
[root@bb2369627996 /]# ls
bin dev etc home lib lib64 lost+found media mnt opt proc root run
sbin srv sys test01 tmp usr var
[root@bb2369627996 /]#

```

## Mysql容器数据共享

```

[root@hoyin test]# docker run -d -p 3310:3306 -v
/home/mysql/conf:/etc/mysql/conf.d -v /home/mysql/data:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=123456 --name mysql01 mysql:5.7
2f011b57949081bd0d8e00d2db222aa2992800e9ac7020d46edf49a2a2f767a

```

```

[root@hoyin test]# docker run -d -p 3311:3306 --volumes-from mysql01 -e
MYSQL_ROOT_PASSWORD=123456 --name mysql02 mysql:5.7
2f011b57949081bd0d8e00d2db222aa2992800e9ac7020d46edf49a2a2f767a

```

因为 mysql 有锁表机制，同时间只能启动一个mysql 容器

共享数据卷共用宿主机同一文件夹 1vN（共享）

删除宿主机目录内容 容器都删除

## Docker 网络

### Docker0

每一个安装Docker 都会分配一个虚拟网卡 veth-pair

```
[root@hoyin docker-test-volumes]# ip addr
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 ###本机回环地址
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 00:16:3e:22:53:5c brd ff:ff:ff:ff:ff:ff
    inet 172.26.47.163/20 brd 172.26.47.255 scope global dynamic noprefixroute
eth0 ###阿里云网络地址
        valid_lft 315119431sec preferred_lft 315119431sec
        inet6 fe80::216:3eff:fe22:535c/64 scope link
            valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN group default
    link/ether 02:42:ca:1a:d6:e8 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0 ### Docker0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:caff:fe1a:d6e8/64 scope link
        valid_lft forever preferred_lft forever
```

每新建一个Docker container 都会创建一对虚拟网址

```
[root@hoyin docker-test-volumes]# docker run -d -P --name t01 tomcat # 新建容器
15ff6001fc433129a097d7cc042147739c8f40c56341bb57a018bbf269faba9c
[root@hoyin docker-test-volumes]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS
PORTS         NAMES
15ff6001fc43   tomcat    "catalina.sh run"       13 seconds ago Up 10 seconds
0.0.0.0:49153->8080/tcp, :::49153->8080/tcp   t01
[root@hoyin docker-test-volumes]# docker exec -it 15ff6001fc43 ip addr # 查看新容器网络
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
84: eth0@if85: <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@hoyin docker-test-volumes]# ip addr
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: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 00:16:3e:22:53:5c brd ff:ff:ff:ff:ff:ff
    inet 172.26.47.163/20 brd 172.26.47.255 scope global dynamic noprefixroute
eth0
        valid_lft 315118323sec preferred_lft 315118323sec
    inet6 fe80::216:3eff:fe22:535c/64 scope link
        valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
    link/ether 02:42:ca:1a:d6:e8 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:caff:fe1a:d6e8/64 scope link
        valid_lft forever preferred_lft forever
85: veth9eb2189@if84: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
master docker0 state UP group default
    link/ether da:3e:2f:13:0a:a9 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::d83e:2fff:fe13:aa9/64 scope link
        valid_lft forever preferred_lft forever

###
### 84: eth0@if85: <====> 85: veth9eb2189@if84:
###

```

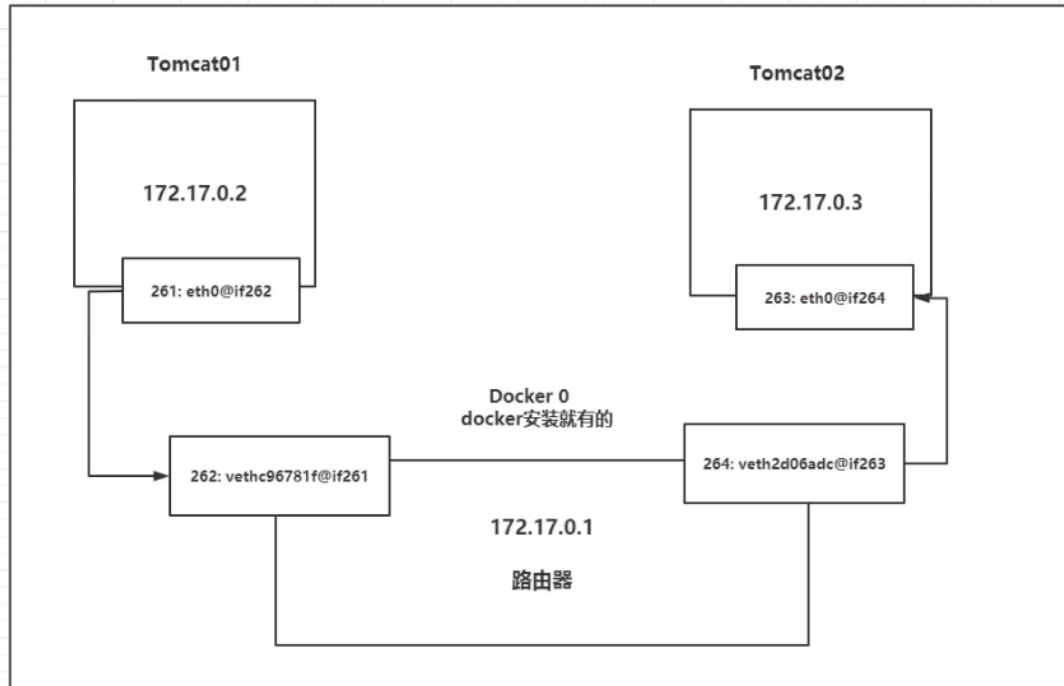
## 容器之间 ip 互通

```

[root@hoyin docker-test-volumes]# docker exec -it t02 ip addr
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
86: eth0@if87: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default
    link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.3/16 brd 172.17.255.255 scope global eth0
        valid_lft forever preferred_lft forever
[root@hoyin docker-test-volumes]# docker exec -it t02 /bin/bash
root@20cfe38b13c0:/usr/local/tomcat# ping 172.17.0.2 # t02 ping t01

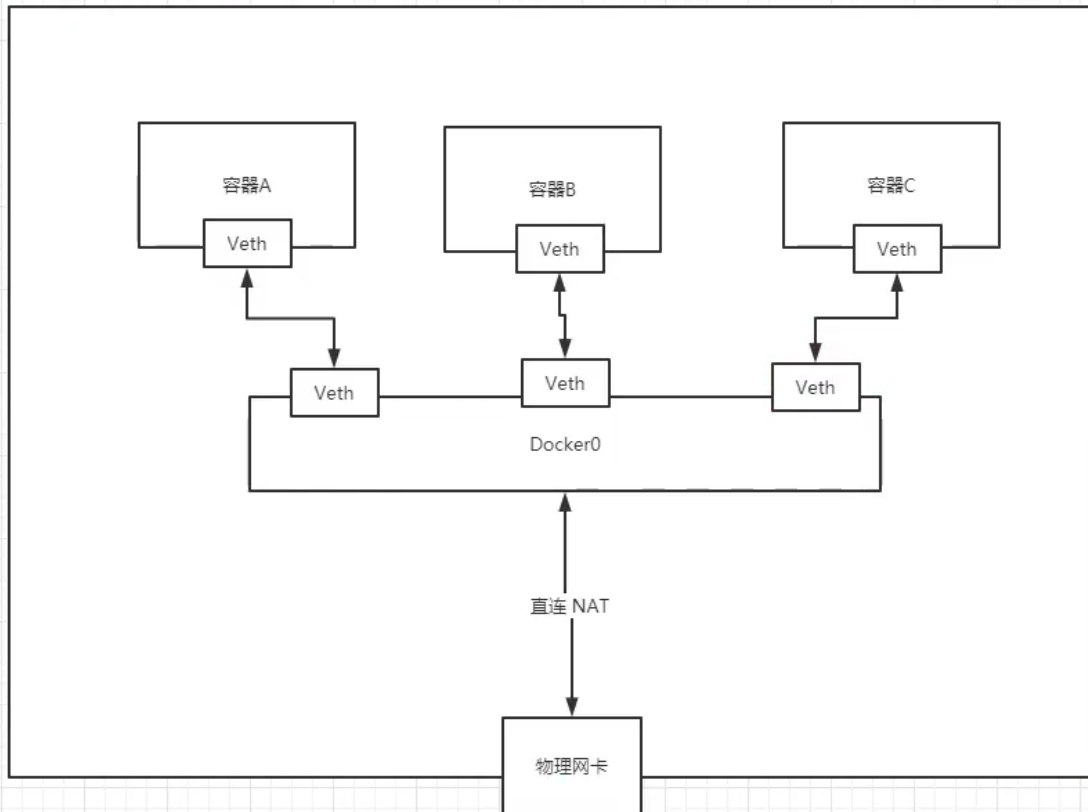
```

```
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.  
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.099 ms  
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.074 ms  
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.073 ms  
###  
### t02 ping t01  
###
```



所有容器不指定网络的情况下，都是docker0路由的，docker会给容器分配个默认的可用ip

## Docker0 与物理网卡是桥接模式



## docker --link

--link 在容器内 /etc/hosts 添加目标地址

```
root@a37db73ce14c:/etc# cat 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
172.17.0.3    t02 20cfe38b13c0
172.17.0.4    a37db73ce14c
```

```
[root@hoyin docker-test-volumes]# docker exec -it t02 ping t01
ping: t01: Name or service not known

[root@hoyin docker-test-volumes]# docker run -d -P --name t03 --link t02 tomcat
a37db73ce14cb07b35f60443f8105ee9b4b11cf9e45b64fd58129405ada26e0f
### 通过--link 通过容器名ping 通
[root@hoyin docker-test-volumes]# docker exec -it t03 ping t02
PING t02 (172.17.0.3) 56(84) bytes of data.
64 bytes from t02 (172.17.0.3): icmp_seq=1 ttl=64 time=0.102 ms
64 bytes from t02 (172.17.0.3): icmp_seq=2 ttl=64 time=0.088 ms
64 bytes from t02 (172.17.0.3): icmp_seq=3 ttl=64 time=0.077 ms
```

# 自定义网络

## 网络模式

bridge: 桥接 (docker 默认)

none: 不配置

host: 和宿主机共享

container: 容器

## docker network ls [查看所有网络]

```
[root@hoyin docker-test-volumes]# docker network ls
NETWORK ID          NAME                DRIVER             SCOPE
9f740ac7996a        bridge             bridge             local
28a223805c7b        host               host               local
f26162d67a66        none               null               local
```

默认配置 dokcer0

```
docker run -d -P --name t01 tomcat
```

```
docker run -d -P --name t01 --net bridge tomcat
```

## docker network create [创建网络]

```
##--driver bridge
##--subnet 192.168.0.0/16
##--gateway 192.168.0.1
#
[root@hoyin docker-test-volumes]# docker network create --driver bridge --subnet
192.168.0.0/16 --gateway 192.168.0.1 mynet
6d3b62f9cf70a6113a41d66fc6cecca0f58df447ef6b9fc507dddba6a7448726
[root@hoyin docker-test-volumes]# docker network ls
NETWORK ID          NAME                DRIVER             SCOPE
9f740ac7996a        bridge             bridge             local
28a223805c7b        host               host               local
6d3b62f9cf70        mynet              bridge             local
f26162d67a66        none               null               local

[root@hoyin docker-test-volumes]# docker network inspect mynet
[
  {
    "Name": "mynet",
    "Id":
"6d3b62f9cf70a6113a41d66fc6cecca0f58df447ef6b9fc507dddba6a7448726",
    "Created": "2021-06-25T10:57:48.535950532+08:00",
    "Scope": "local",
    "Driver": "bridge",
```

```

    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "192.168.0.0/16",
          "Gateway": "192.168.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]

```

根据新建网络创建容器

```

[root@hoyin docker-test-volumes]# docker run -d -P --name mynet01 --net mynet
tomcat
a3df175409f8dda799661beb994946966863a1b33a2b8e6b51c7514d9af0b709
[root@hoyin docker-test-volumes]# docker run -d -P --name mynet02 --net mynet
tomcat
1d1a2e452f49f53265864687c8dbb6255e31af292493a27ba9dde7af41263caa
[root@hoyin docker-test-volumes]# docker network inspect mynet
[
  {
    "Name": "mynet",
    "Id":
"6d3b62f9cf70a6113a41d66fc6cecca0f58df447ef6b9fc507dddba6a7448726",
    "Created": "2021-06-25T10:57:48.535950532+08:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "192.168.0.0/16",
          "Gateway": "192.168.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {

```

```

        "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
        "1d1a2e452f49f53265864687c8dbb6255e31af292493a27ba9dde7af41263caa":
    {
            "Name": "mynet02",
            "EndpointID":
"9fcaa406f90c3cc8e53db7964a894db4a04b6abfc035e38d8a4921f3d9c0038d",
            "MacAddress": "02:42:c0:a8:00:03",
            "IPv4Address": "192.168.0.3/16",
            "IPv6Address": ""
        },
        "a3df175409f8dda799661beb994946966863a1b33a2b8e6b51c7514d9af0b709":
    {
            "Name": "mynet01",
            "EndpointID":
"f23334eb17933c81f9c7934ddf7de5a3376ca3e43393e1771e45b931f4f095f3",
            "MacAddress": "02:42:c0:a8:00:02",
            "IPv4Address": "192.168.0.2/16",
            "IPv6Address": ""
        }
    },
    "Options": {},
    "Labels": {}
}
]

```

### 新建的容器网络地址 在 [Containers]

```

[root@hoyin docker-test-volumes]# docker exec -it mynet01 /bin/bash
root@a3df175409f8:/usr/local/tomcat#
root@a3df175409f8:/usr/local/tomcat# ping mynet02
PING mynet02 (192.168.0.3) 56(84) bytes of data.
64 bytes from mynet02.mynet (192.168.0.3): icmp_seq=1 ttl=64 time=0.082 ms
64 bytes from mynet02.mynet (192.168.0.3): icmp_seq=2 ttl=64 time=0.067 ms

```

## 网络联通

### docker network connect [OPTIONS] NETWORK CONTAINER

```

[root@hoyin docker-test-volumes]# docker run -d -P --name t01 tomcat
4f416f9b98b1d0cb9b1171e262f2af0daf44a3fa3a9ea38cb1ae9a8d81c05c2c
[root@hoyin docker-test-volumes]# docker network connect mynet t01
[root@hoyin docker-test-volumes]# docker network inspect mynet
[
    {
        "Name": "mynet",
        "Id":
"6d3b62f9cf70a6113a41d66fc6cecca0f58df447ef6b9fc507dddba6a7448726",
        "Created": "2021-06-25T10:57:48.535950532+08:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
    }
]

```

```

    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "192.168.0.0/16",
          "Gateway": "192.168.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "1d1a2e452f49f53265864687c8dbb6255e31af292493a27ba9dde7af41263caa": {
        "Name": "mynet02",
        "EndpointID":
"9fcaa406f90c3cc8e53db7964a894db4a04b6abfc035e38d8a4921f3d9c0038d",
        "MacAddress": "02:42:c0:a8:00:03",
        "IPv4Address": "192.168.0.3/16",
        "IPv6Address": ""
      },
      "4f416f9b98b1d0cb9b1171e262f2af0daf44a3fa3a9ea38cb1ae9a8d81c05c2c": {
        "Name": "t01",
        "EndpointID":
"a20a8e0f82fa23ab60b93ea096ed93c52836708a1dece19a713016fbe6d7b7a5",
        "MacAddress": "02:42:c0:a8:00:04",
        "IPv4Address": "192.168.0.4/16",
        "IPv6Address": ""
      },
      "a3df175409f8dda799661beb994946966863a1b33a2b8e6b51c7514d9af0b709": {
        "Name": "mynet01",
        "EndpointID":
"f23334eb17933c81f9c7934ddf7de5a3376ca3e43393e1771e45b931f4f095f3",
        "MacAddress": "02:42:c0:a8:00:02",
        "IPv4Address": "192.168.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]

```

```
[root@hoyin docker-test-volumes]# docker network create redis --subnet
192.169.0.0/16
e219f39a0cbd34605878b7ff35e7c5829444a90f96042e7cc41b2d692f8ac9c4

for port in $(seq 1 6); \
do \
mkdir -p /mydata/redis/node- $\{port\}$ /conf
touch /mydata/redis/node- $\{port\}$ /conf/redis.conf
cat << EOF >/mydata/redis/node- $\{port\}$ /conf/redis.conf
port 6379
bind 0.0.0.0
cluster-enabled yes
cluster-config-file nodes.conf
cluster-node-timeout 5000
cluster-announce-ip 192.169.0.1 $\{port\}$ 
cluster-announce-port 6379
cluster-announce-bus-port 16379
appendonly yes
EOF
done

for port in $(seq 1 6); \
do \
docker run -p 637 $\{port\}$ :6379 -p 1637 $\{port\}$ :16379 --name redis0 $\{port\}$  -v
/mydata/redis/node- $\{port\}$ /data:/data -v
/mydata/redis/node- $\{port\}$ /conf/redis.conf:/etc/redis/redis.conf -d --net redis
--ip 192.169.0.1 $\{port\}$  redis:5.0.9-alpine3.11 redis-server
/etc/redis/redis.conf
done

docker run -d -p 6371:6379 -p 9371:6379 --name redis01 -v /mydata/redis/node-
1/data:/data -v /mydata/redis/node-1/conf/redis.conf:/etc/redis/redis.conf --net
redis --ip 192.169.0.11 redis:5.0.9-alpine3.11 redis-server
/etc/redis/redis.conf

docker run -d -p 6372:6379 -p 9372:6379 --name redis02 -v /mydata/redis/node-
2/data:/data -v /mydata/redis/node-2/conf/redis.conf:/etc/redis/redis.conf --net
redis --ip 192.169.0.12 redis:5.0.9-alpine3.11 redis-server
/etc/redis/redis.conf

docker run -d -p 6373:6379 -p 9373:6379 --name redis03 -v /mydata/redis/node-
3/data:/data -v /mydata/redis/node-3/conf/redis.conf:/etc/redis/redis.conf --net
redis --ip 192.169.0.13 redis:5.0.9-alpine3.11 redis-server
/etc/redis/redis.conf

docker run -d -p 6374:6379 -p 9374:6379 --name redis04 -v /mydata/redis/node-
4/data:/data -v /mydata/redis/node-4/conf/redis.conf:/etc/redis/redis.conf --net
redis --ip 192.169.0.14 redis:5.0.9-alpine3.11 redis-server
/etc/redis/redis.conf

docker run -d -p 6375:6379 -p 9375:6379 --name redis05 -v /mydata/redis/node-
5/data:/data -v /mydata/redis/node-5/conf/redis.conf:/etc/redis/redis.conf --net
redis --ip 192.169.0.15 redis:5.0.9-alpine3.11 redis-server
/etc/redis/redis.conf
```



```

docker run -d -p 6376:6379 -p 9376:6379 --name redis06 -v /mydata/redis/node-6/data: data -v /mydata/redis/node-6/conf/redis/conf:/etc/redis/redis.conf --net redis --ip 192.169.0.16 redis:5.0.9-alpine3.11 redis-server /etc/redis/redis.conf

redis-cli --cluster create 192.169.0.11:6379 192.169.0.12:6379 192.169.0.13:6379 192.169.0.14:6379 192.169.0.15:6379 192.169.0.16:6379 --cluster-replicas 1

[root@hoyin conf]# for port in $(seq 1 6); \
> do \
> mkdir -p /mydata/redis/node- $\{port\}$ /conf
> touch /mydata/redis/node- $\{port\}$ /conf/redis.conf
> cat << EOF >/mydata/redis/node- $\{port\}$ /conf/redis.conf
> port 6379
> bind 0.0.0.0
> cluster-enabled yes
> cluster-config-file nodes.conf
> cluster-node-timeout 5000
> cluster-announce-ip 192.169.0.1 $\{port\}$ 
> cluster-announce-port 6379
> cluster-announce-bus-port 16379
> appendonly yes
> EOF
> done
[root@hoyin conf]# for port in $(seq 1 6); \
> do \
> docker run -p 637 $\{port\}$ :6379 -p 1637 $\{port\}$ :16379 --name redis0 $\{port\}$  -v /mydata/redis/node- $\{port\}$ /data:/data -v /mydata/redis/node- $\{port\}$ /conf/redis.conf:/etc/redis/redis.conf -d --net redis --ip 192.169.0.1 $\{port\}$  redis:5.0.9-alpine3.11 redis-server /etc/redis/redis.conf
> done
3a499e7b402396938927419535be81ede045e25e9c1d97bb74ba47b5a9ff03a8
63d72f07d16edb3f84e44b0ca335e7eedda27fe31179153cba1b584cd99689ad
31b2e36b5bcd67b75e09bf375440598f26bcdff2c86e309c63870aa7a268b58b
34b09b62a1aa5023564d2af5f8fcd4067fa389adf6f09b82e76a67a2ea331e90
8757b9950192caf2b8c370ac58298277a0cbec3bd66757743bb9fb0ef84804ea
6001ceca4e26b028856e598290a10e57c32dfde96d7cc0a3c41f909c9abdca71
[root@hoyin conf]# docker exec -it redis01 /bin/sh
/data #
/data #
/data # redis-cli --cluster create 192.169.0.11:6379 192.169.0.12:6379 192.169.0.13:6379 192.169.0.14:6379 192.169.0.15:6379 192.169.0.16:6379 --cluster-replicas 1
>>> Performing hash slots allocation on 6 nodes...
Master[0] -> Slots 0 - 5460
Master[1] -> Slots 5461 - 10922
Master[2] -> Slots 10923 - 16383
Adding replica 192.169.0.15:6379 to 192.169.0.11:6379
Adding replica 192.169.0.16:6379 to 192.169.0.12:6379
Adding replica 192.169.0.14:6379 to 192.169.0.13:6379
M: f345e1c37262b4b0ae7b6fdef947676adc2deace 192.169.0.11:6379
slots:[0-5460] (5461 slots) master
M: 52b778473a0c0ae59bf4b7a87ff2a3334072bf8c 192.169.0.12:6379
slots:[5461-10922] (5462 slots) master

```

```

M: cc5b1f4f11732a6b1d408a074d0b7f224ccafb9c 192.169.0.13:6379
  slots:[10923-16383] (5461 slots) master
S: f4876241b5f7f4ae8b2b15f1d8b6e2043877b194 192.169.0.14:6379
  replicates cc5b1f4f11732a6b1d408a074d0b7f224ccafb9c
S: d1bf8c446cc8594f5fdc174ae98b2867b41ccd10 192.169.0.15:6379
  replicates f345e1c37262b4b0ae7b6fdef947676adc2deace
S: 3e4b623d2dac8f0d02f614f23fac1789cace8022 192.169.0.16:6379
  replicates 52b778473a0c0ae59bf4b7a87ff2a3334072bf8c
Can I set the above configuration? (type 'yes' to accept): yes
>>> Nodes configuration updated
>>> Assign a different config epoch to each node
>>> Sending CLUSTER MEET messages to join the cluster
waiting for the cluster to join
...
>>> Performing cluster check (using node 192.169.0.11:6379)
M: f345e1c37262b4b0ae7b6fdef947676adc2deace 192.169.0.11:6379
  slots:[0-5460] (5461 slots) master
  1 additional replica(s)
S: f4876241b5f7f4ae8b2b15f1d8b6e2043877b194 192.169.0.14:6379
  slots: (0 slots) slave
  replicates cc5b1f4f11732a6b1d408a074d0b7f224ccafb9c
S: d1bf8c446cc8594f5fdc174ae98b2867b41ccd10 192.169.0.15:6379
  slots: (0 slots) slave
  replicates f345e1c37262b4b0ae7b6fdef947676adc2deace
S: 3e4b623d2dac8f0d02f614f23fac1789cace8022 192.169.0.16:6379
  slots: (0 slots) slave
  replicates 52b778473a0c0ae59bf4b7a87ff2a3334072bf8c
M: 52b778473a0c0ae59bf4b7a87ff2a3334072bf8c 192.169.0.12:6379
  slots:[5461-10922] (5462 slots) master
  1 additional replica(s)
M: cc5b1f4f11732a6b1d408a074d0b7f224ccafb9c 192.169.0.13:6379
  slots:[10923-16383] (5461 slots) master
  1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.

```

```

/data # redis-cli -c
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> set a b
192.169.0.13:6379> cluster info
cluster_state:ok
cluster_slots_assigned:16384
cluster_slots_ok:16384
cluster_slots_pfail:0
cluster_slots_fail:0
cluster_known_nodes:6
cluster_size:3
cluster_current_epoch:6
cluster_my_epoch:3
cluster_stats_messages_ping_sent:845
cluster_stats_messages_pong_sent:890
cluster_stats_messages_meet_sent:3
cluster_stats_messages_sent:1738
cluster_stats_messages_ping_received:888

```

```

cluster_stats_messages_pong_received:848
cluster_stats_messages_meet_received:2
cluster_stats_messages_received:1738
192.169.0.13:6379> cluster nodes
52b778473a0c0ae59bf4b7a87ff2a3334072bf8c 192.169.0.12:6379@16379 master - 0
1624607469617 2 connected 5461-10922
3e4b623d2dac8f0d02f614f23fac1789cace8022 192.169.0.16:6379@16379 slave
52b778473a0c0ae59bf4b7a87ff2a3334072bf8c 0 1624607468613 6 connected
f4876241b5f7f4ae8b2b15f1d8b6e2043877b194 192.169.0.14:6379@16379 slave
cc5b1f4f11732a6b1d408a074d0b7f224ccafb9c 0 1624607469000 4 connected
cc5b1f4f11732a6b1d408a074d0b7f224ccafb9c 192.169.0.13:6379@16379 myself,master -
0 1624607467000 3 connected 10923-16383
f345e1c37262b4b0ae7b6fdef947676adc2deace 192.169.0.11:6379@16379 master - 0
1624607469000 1 connected 0-5460
d1bf8c446cc8594f5fdc174ae98b2867b41ccd10 192.169.0.15:6379@16379 slave
f345e1c37262b4b0ae7b6fdef947676adc2deace 0 1624607468512 5 connected

#关闭 redis03
192.169.0.14:6379> cluster nodes
f345e1c37262b4b0ae7b6fdef947676adc2deace 192.169.0.11:6379@16379 master - 0
1624607671552 1 connected 0-5460
cc5b1f4f11732a6b1d408a074d0b7f224ccafb9c 192.169.0.13:6379@16379 master,fail -
1624607631340 1624607630535 3 connected
f4876241b5f7f4ae8b2b15f1d8b6e2043877b194 192.169.0.14:6379@16379 myself,master -
0 1624607670000 7 connected 10923-16383
d1bf8c446cc8594f5fdc174ae98b2867b41ccd10 192.169.0.15:6379@16379 slave
f345e1c37262b4b0ae7b6fdef947676adc2deace 0 1624607671000 5 connected
52b778473a0c0ae59bf4b7a87ff2a3334072bf8c 192.169.0.12:6379@16379 master - 0
1624607671552 2 connected 5461-10922
3e4b623d2dac8f0d02f614f23fac1789cace8022 192.169.0.16:6379@16379 slave
52b778473a0c0ae59bf4b7a87ff2a3334072bf8c 0 1624607671952 6 connected
192.169.0.14:6379> cluster nodes
f345e1c37262b4b0ae7b6fdef947676adc2deace 192.169.0.11:6379@16379 master - 0
1624607710239 1 connected 0-5460
cc5b1f4f11732a6b1d408a074d0b7f224ccafb9c 192.169.0.13:6379@16379 slave
f4876241b5f7f4ae8b2b15f1d8b6e2043877b194 0 1624607709233 7 connected
f4876241b5f7f4ae8b2b15f1d8b6e2043877b194 192.169.0.14:6379@16379 myself,master -
0 1624607709000 7 connected 10923-16383
d1bf8c446cc8594f5fdc174ae98b2867b41ccd10 192.169.0.15:6379@16379 slave
f345e1c37262b4b0ae7b6fdef947676adc2deace 0 1624607709000 5 connected
52b778473a0c0ae59bf4b7a87ff2a3334072bf8c 192.169.0.12:6379@16379 master - 0
1624607709000 2 connected 5461-10922
3e4b623d2dac8f0d02f614f23fac1789cace8022 192.169.0.16:6379@16379 slave
52b778473a0c0ae59bf4b7a87ff2a3334072bf8c 0 1624607710038 6 connected

```

# Docker compose

## 安装 docker-compose

```
[root@hoyin ~]# curl -L
https://get.daocloud.io/docker/compose/releases/download/1.25.5/docker-compose-
`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
curl -I https : //get.daocloud.io/docker/compose/releases /download/1. 25 .
5/docker-compose-`uname -s ` -uname -mT > /usr/local/bin/docker-compose
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 423 100 423 0 0 100 0 0:00:04 0:00:04 --:--:-- 100
100 16.7M 100 16.7M 0 0 3408k 0 0:00:05 0:00:05 --:--:-- 5127k

[root@hoyin bin]# sudo chmod +x /usr/local/bin/docker-compose #授权
[root@hoyin bin]# docker-compose version
docker-compose version 1.25.5, build 8a1c60f6
docker-py version: 4.1.0
CPython version: 3.7.5
OpenSSL version: OpenSSL 1.1.0f 10 Sep 2019
```

```
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello world! I have been seen {} times.\n'.format(count)
```

## yaml

```
version: ''
services:
    服务1:
    服务2:
# 其他配置
```

