```

```

Docker

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

```
systemctl start 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
b8dfde127a29: 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

删除 资源
rm -rf /var/lib/docker

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

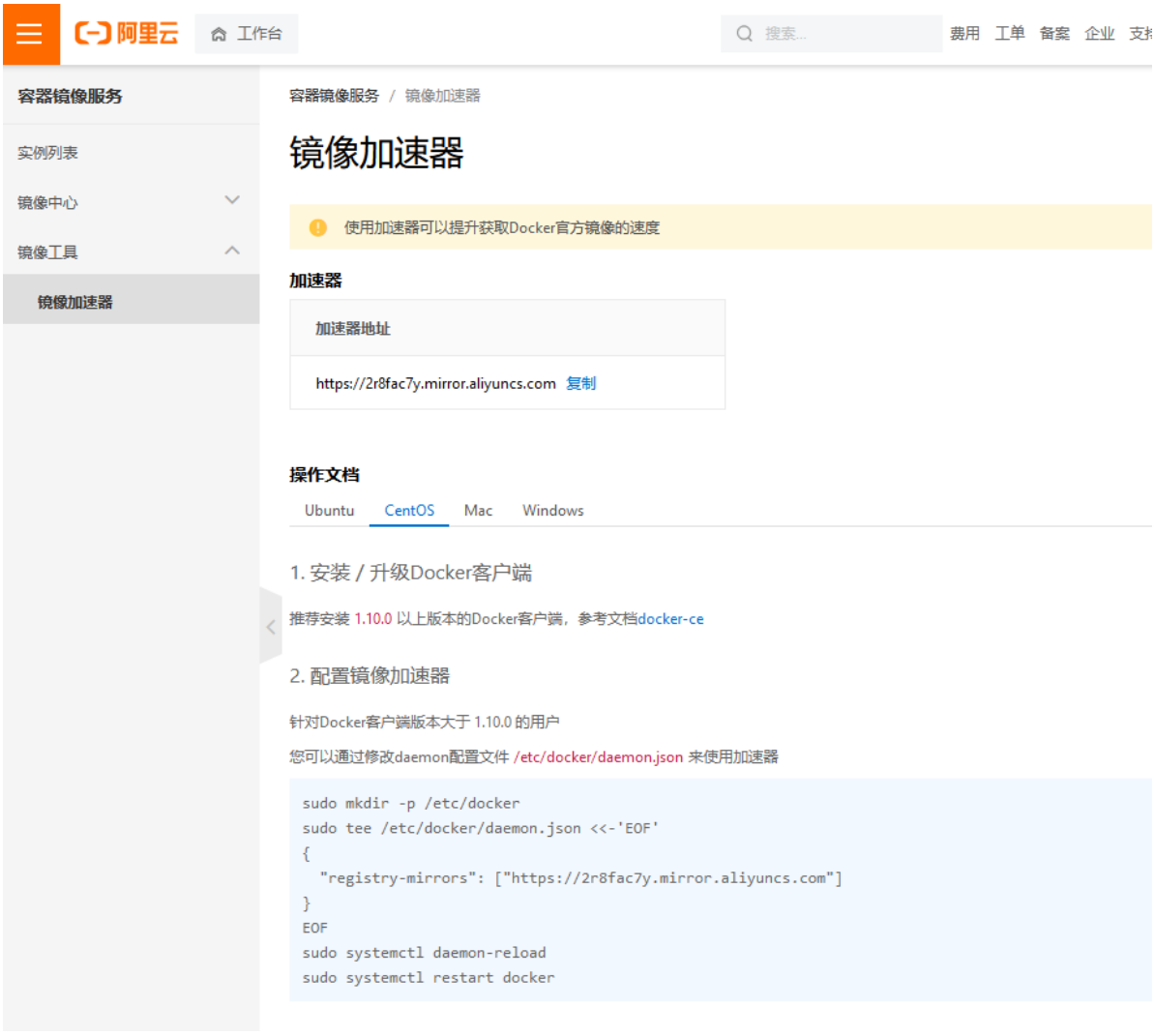
配置阿里云镜像加速

阿里云

容器镜像服务

镜像工具

镜像加速器



配置使用

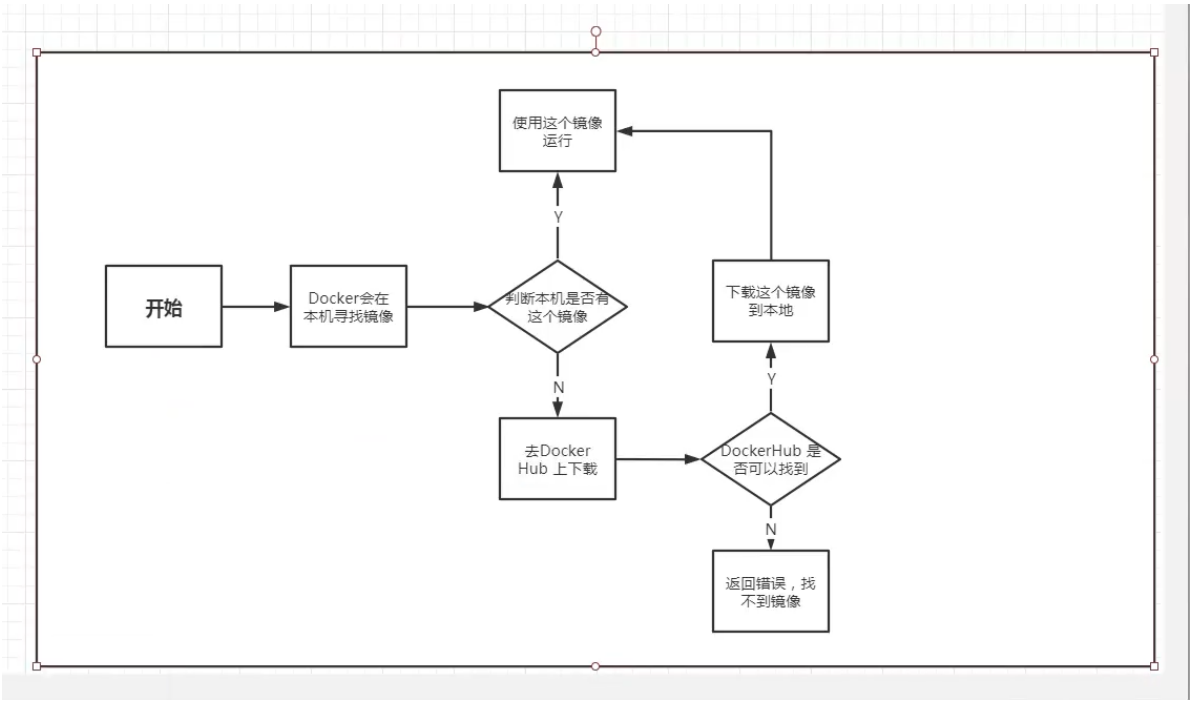
```
sudo mkdir -p /etc/docker

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

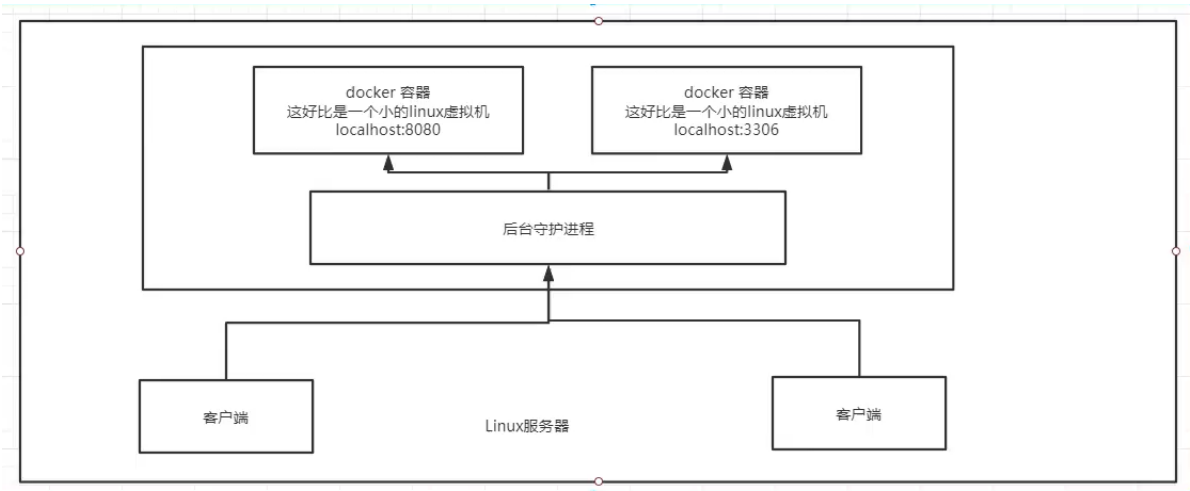


底层原理

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

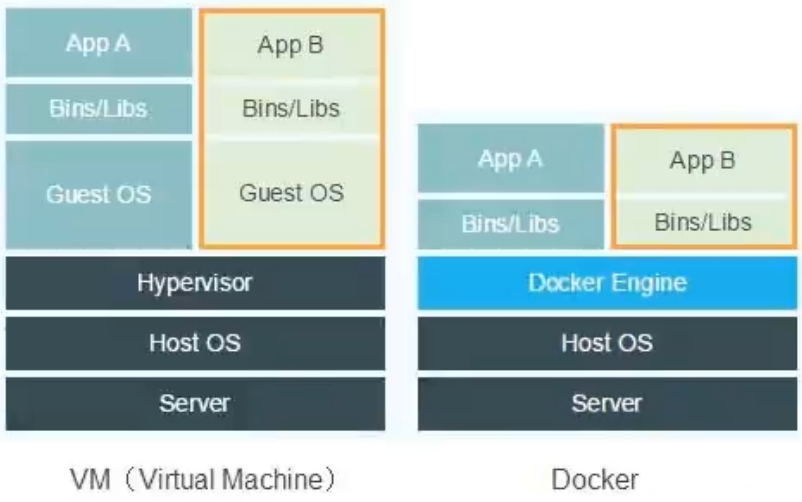
1 .Docker有着比虚拟机更少的抽象层

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纵系统。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 dockerId

```
[root@hoyin ~]# docker rmi -f 2c9028880e58
Untagged: mysql:5.7
Untagged:
mysql@sha256:a682e3c78fc5bd941e9db080b4796c75f69a28a8cad65677c23f7a9f18ba21fa
Deleted: sha256:2c9028880e5814e8923c278d7e2059f9066d56608a21cd3f83a01e3337bacd68
Deleted: sha256:c49c5c776f1bc87cdfff451ef39ce16a1ef45829e10203f4d9a153a6889ec15e
Deleted: sha256:8345316eca77700e624706114465291135797l2a787d356e5c8656a41c244aee
Deleted: sha256:8ae51b87111404bd3e3bde4115ea2fe3fd2bb2cf67158460423c361a24df156b
Deleted: sha256:9d5afda6f6dcf8dd59aef5c02099f1d3b3b0c9ae4f2bb7a61627613e8cdfe562
```

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
```

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

列出所有运行的容器 docker ps

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

退出容器 exit

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

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
```

启动和停止容器

```
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

后台启动容器 docker run -d 镜像名

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
```

查看logs docker 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
```

查看容器中的进程信息 docker top

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
```

查看镜像的元数据 docker inspect

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/e5c0e34be3611d3c2b0e121262372243a23b46c5b5e540760208
bb747c5c26b0/resolv.conf",
```

```
        "HostnamePath":
"/var/lib/docker/containers/e5c0e34be3611d3c2b0e121262372243a23b46c5b5e540760208
bb747c5c26b0/hostname",
        "HostsPath":
"/var/lib/docker/containers/e5c0e34be3611d3c2b0e121262372243a23b46c5b5e540760208
bb747c5c26b0/hosts",
        "LogPath":
"/var/lib/docker/containers/e5c0e34be3611d3c2b0e121262372243a23b46c5b5e540760208
bb747c5c26b0/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": "",
            "UsernsMode": "",
            "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/overlay2/9b7b52784a0aaddcd63a43094d571b13d4f6f34ff842f9141aba6d
1e516a10c8-
init/diff:/var/lib/docker/overlay2/9d30ff772de9466d3dc90840317269af610265c3a04bc
ebdccd5c3b365818e5e/diff",
                "MergedDir":
"/var/lib/docker/overlay2/9b7b52784a0aaddcd63a43094d571b13d4f6f34ff842f9141aba6d
1e516a10c8/merged",
                "UpperDir":
"/var/lib/docker/overlay2/9b7b52784a0aaddcd63a43094d571b13d4f6f34ff842f9141aba6d
1e516a10c8/diff",
                "WorkDir":
"/var/lib/docker/overlay2/9b7b52784a0aaddcd63a43094d571b13d4f6f34ff842f9141aba6d
1e516a10c8/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
                }
            }
        }
    }
]
```

进入当前正在运行的容器 exec attach

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
```

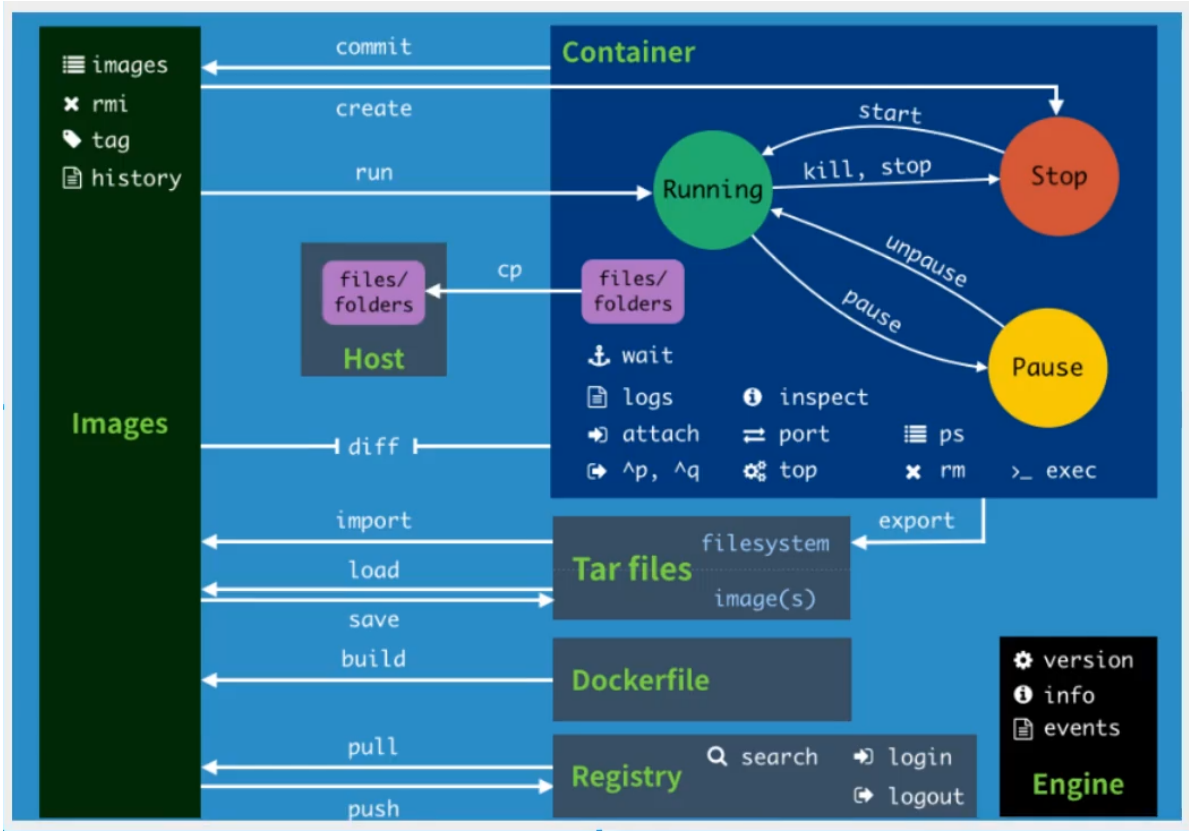
```
#docker exec    开启新得终端        exit 退出后不停止容器
#docker attach  正在运行的终端      exit 退出后直接停止容器
```

从容器内拷贝文件到主机 docker cp

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
```

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                              DESCRIPTION
  STARS     OFFICIAL   AUTOMATED
nginx                             Official build of Nginx.
15034     [OK]
jwilder/nginx-proxy               Automated Nginx reverse proxy for docker con…
  2038                [OK]
richarvey/nginx-php-fpm           Container running Nginx + PHP-FPM capable of…
  814                 [OK]
jc21/nginx-proxy-manager          Docker container for managing Nginx proxy ho…
  199
```

```
linuxserver/nginx                   An Nginx container, brought to you by LinuxS…
  147
tiangolo/nginx-rtmp                 Docker image with Nginx using the nginx-rtmp…
  130                   [OK]
jlesage/nginx-proxy-manager         Docker container for Nginx Proxy Manager
118                   [OK]
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

```
[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>
```

阿里云端口设置

```
[root@hoyin ~]# firewall-cmd --list-ports
FirewallD is not running
[root@hoyin ~]# systemctl start firewalld

[root@hoyin ~]# firewall-cmd --permanent --zone=public --add-port=3344/tcp
success
[root@hoyin ~]# firewall-cmd --permanent --zone=public --add-port=80/tcp
success
[root@hoyin ~]# firewall-cmd --reload
success
[root@hoyin ~]# firewall-cmd --list-ports
3344/tcp 80/tcp
[root@hoyin ~]#
```

```
查看已开放的端口
firewall-cmd --list-ports
开放端口(开放后需要要重启防火墙才生效)firewall-cmd --zone=public --add-port=3338/tcp --permanent
重启防火墙
firewall-cmd --reload
关闭端口(关闭后需要要重启防火墙才生效)firewall-cmd --zone=public --remove-port=3338/tcp --permanent
开机启动防火墙
systemctl enable firewalld开启防火墙
systemctl start firewalld禁止防火墙开机启动
systemctl disable firewalld停止防火墙
systemctl stop firewalld
```