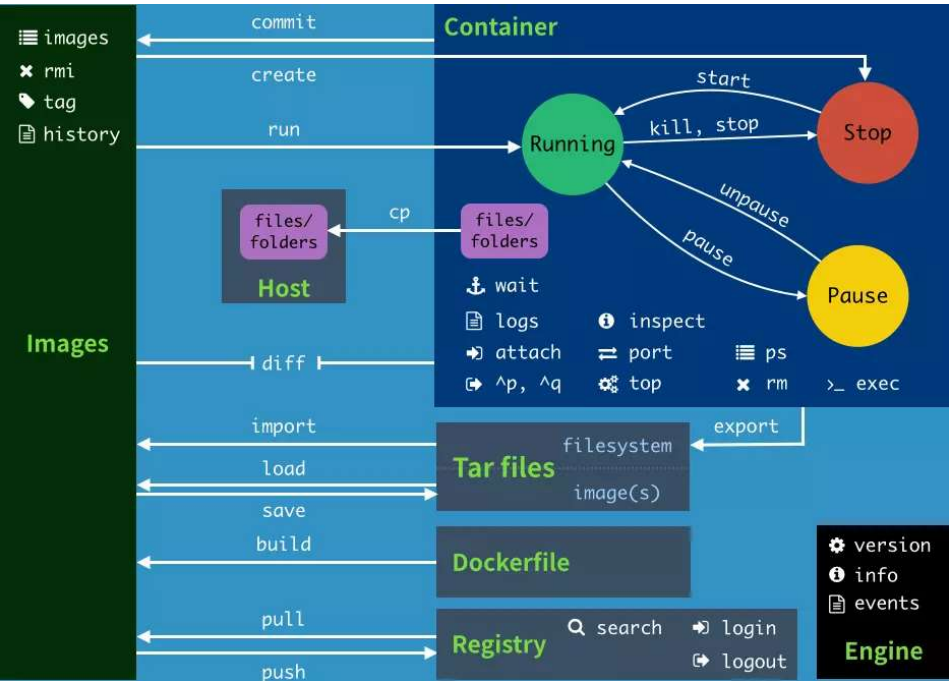


Docker命令

17 2019.04.18 18:43:20 字数 400 阅读 2406



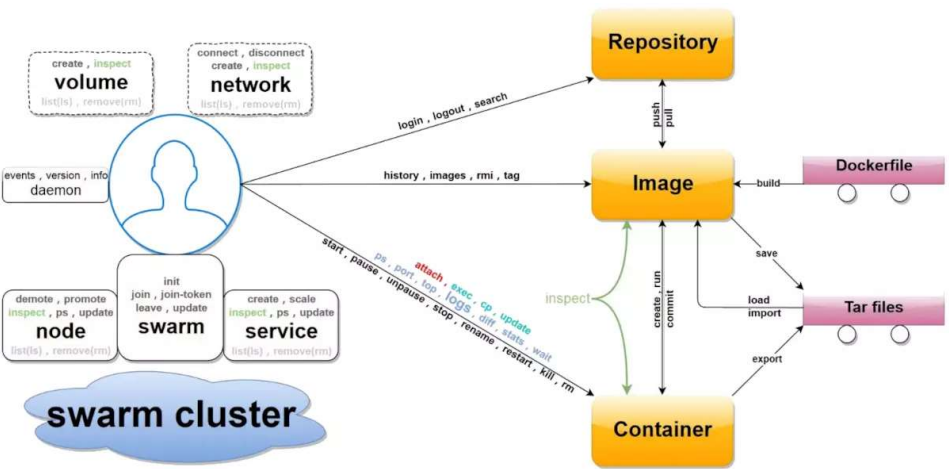
Docker命令

参考资料

- Docker文档 <https://docs.docker.com>

命令格式

1 | docker <选项><命令><参数>



Docker命令


选项说明

- [] 表示设置选项时可以设置不同的值，并且可以多次使用。

写下你的评论...

评论6 赞85 ...

广告

 JunChow520

拥有857钻 (约146.92元)

关注

PHP二维码
阅读 124

Laya加载器
阅读 5

LayaAir笔记
阅读 1

精彩继续

高情商马伊琍也踩坑：
这是无数人犯过的大忌
阅读 13605



Redis单线程？别逗了，Redis6.0多线程
重磅来袭！
阅读 2029

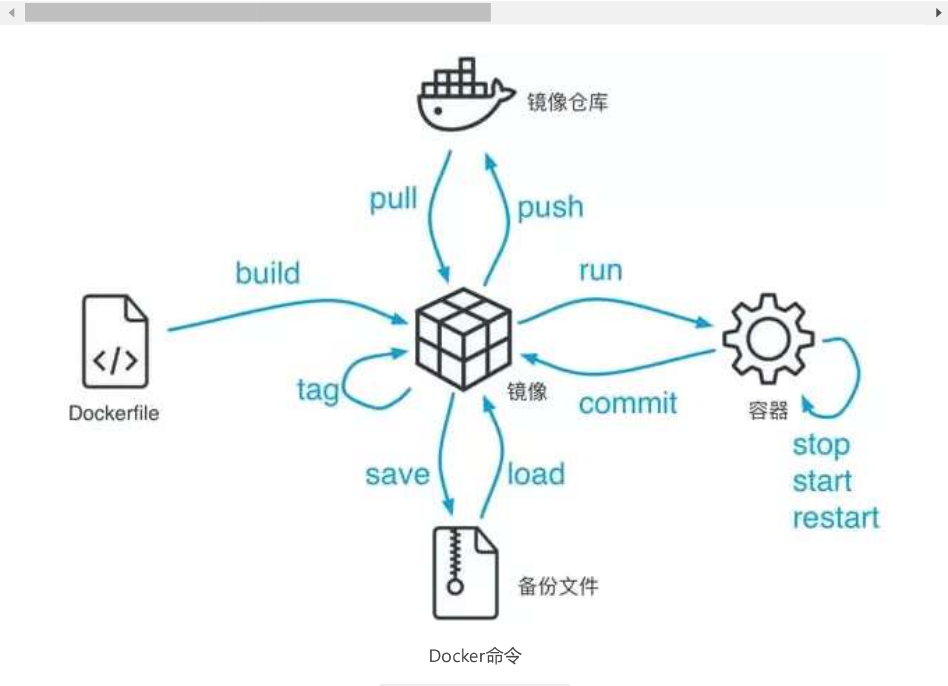


广告

```
1 $ docker --help
2
3 用法:  docker COMMAND
4
5 A self-sufficient runtime for containers
6
7 选项:
8     --config string      Location of client config files (default
9                          "C:\\Users\\junchow\\.docker")
10
11 -D, --debug             开启debug模式
12 -H, --host list         Daemon socket(s) to connect to
13 -l, --log-level string   设置日志级别, 级别分为debug|info|warn|error|fatal, 默认为info。
14     --tls               Use TLS; implied by --tlsverify
15     --tlscacert string   Trust certs signed only by this CA (default
16                          "C:\\Users\\junchow\\.docker\\machine\\machines\\default\\ca.pem
17     --tlscert string     Path to TLS certificate file (default
18                          "C:\\Users\\junchow\\.docker\\machine\\machines\\default\\cert.p
19     --tlskey string      Path to TLS key file (default
20                          "C:\\Users\\junchow\\.docker\\machine\\machines\\default\\key.pe
21     --tlsverify          Use TLS and verify the remote (default true)
22 -v, --version           终端打印显示版本信息并退出
23
24 管理命令:
25     config      管理Docker配置
26     container   管理容器
27     image       管理镜像
28     network     管理网络
29     node        管理Swarm节点
30     plugin      管理插件
31     secret      管理Docker安全
32     service     管理服务
33     swarm       管理Swarm集群
34     system      管理Docker系统
35     trust       Manage trust on Docker images
36     volume      管理卷
37
38 命令:
39     attach      将标准输入和标准输出连接到正在运行的容器
40     build       使用dockerfile文件创建镜像
41     commit      从容器的修改项中创建新的镜像
42     cp         将容器的目录或文件复制到本地文件系统中
43     create      创建一个新的镜像
44     diff        检查容器文件系统的修改
45     events      实时输出docker服务器中发生的事件
46     exec        从外部运行容器内部的命令
47     export      将容器的文件系统到处为tar文件包
48     history     显示镜像的历史
49     images      输出镜像列表
50     import      从压缩为tar文件的文件系统中创建镜像
51     info        显示当前系统信息、docker容器与镜像个数、设置信息等
52     inspect     使用JSON格式显示容器与镜像的详细信息
53     kill        向容器发送kill信号关闭容器
54     load        从tar文件或标准输入中加载镜像
55     login       登录docker注册服务器
56     logout      退出docker注册服务器
57     logs        输出容器日志信息
58     pause       暂停容器中正在运行的所有进程
59     port        查看容器的端口是否处于开放状态
60     ps          输出容器列表
61     pull        从注册服务器中拉取一个镜像或仓库
62     push        将镜像推送到docker注册服务器
63     rename      重命名一个容器
64     restart     重启一个或多个容器
65     rm          删除一个或多个容器, 若没有指定标签则删除latest标签。
66     rmi         删除一个或多个镜像, 若没有指定标签则删除latest标签。
67     run         在一个新容器中运行命令, 用于指定镜像创建容器。
68     save        将一个或多个镜像保存为tar包
69     search      从Docker Hub中搜索镜像
70     start       启动一个或多个已经停止的容器
71     stats       Display a live stream of container(s) resource usage statistics
72     stop        停止一个或多个正在运行的容器
73
74
```



Run 'docker COMMAND --help' for more information on a command.



广告

容器生命周期

docker run

```
1 # run用于指定镜像创建容器
2 $ docker run [选项] <镜像名称, id> [命令] [参数]
3
4 # 选项
5 -d, --detach=false 指定容器运行于前台还是后台, 默认为false。
6 -i, --interactive=false 打开标准输入用于控制台交互
7 -t, --tty=false 分配tty设备用来支持终端登录, 默认为false。
8 -u, --user="" 指定容器的用户
9 -a, --attach=[] 登录容器, 必须是以docker run -d启动的容器。
10 -w, --workdir="" 设置容器的工作目录
11 -c, --cpu-shares=0 设置容器CPU权重, 在CPU共享场景下使用。
12 -e, --env=[] 设置环境变量, 容器中可使用该环境变量。
13 -m, --memory="" 设置容器的内存上限
14 -p, --public=[] 设置容器暴露的端口
15 -h, --hostname="" 设置容器的主机名
16 -v, --volume=[] 设置容器挂载的存储卷, 也就是挂载到容器的某个目录。
17 --volum-from=[] 给容器挂载其他容器上的卷, 也就是挂载到容器的某个目录。
18 --cap-add=[] 添加权限
19 --cap-drop=[] 删除权限
20 --cidfile="" 运行容器后在指定文件中写入容器PID值, 这是典型的监控系统的用法。
21 --cpuset="" 设置容器可以使用那些CPU, 此参数用来设置容器独占CPU。
22 --device=[] 添加主机设备给容器, 相当于设备直通。
23 --dns=[] 设置容器的DNS服务器
24 --dns-search=[] 设置容器的DNS搜索域名, 写入到容器的/etc/resolv.conf文件。
25 --env-file=[] 设置环境变量文件, 文件格式为每行一个环境变量。
26 --expose=[] 设置容器暴露的端口, 即修改镜像的暴露端口。
27 --link=[] 设置容器之间的关联关系, 使用其他容器的IP、env等信息。
28 --lxc-conf=[] 设置容器的配置文件, 只有在指定--exe-driver=lxc时使用。
29 --name="" 设置容器的名称, 可通过名字进行容器管理, links特性需要使用名字。
30 --net="bridge" 容器网络设置
31 --privileged=false 设置容器是否为特权容器, 特权容器拥有所有的capabilities。
32 --restart="no" 设置让其停止后的重启策略
33 --rm=false 设置容器停止后自动删除容器, 不支持以docker run -d启动的容器。
34 --sig-proxy=true 设置由代理接收并处理信号, 但SIGCHLD、SIGSTOP、SIGKILL不能被代理。
35
36 # 命令
37
38 -d, --detach Detach模式, 默认为守护进程模式, 即容器以后台方式运行。
39 --rm=false 若容器内的进程终止则自动删除容器, 禁止和-d选项一起使用。
```

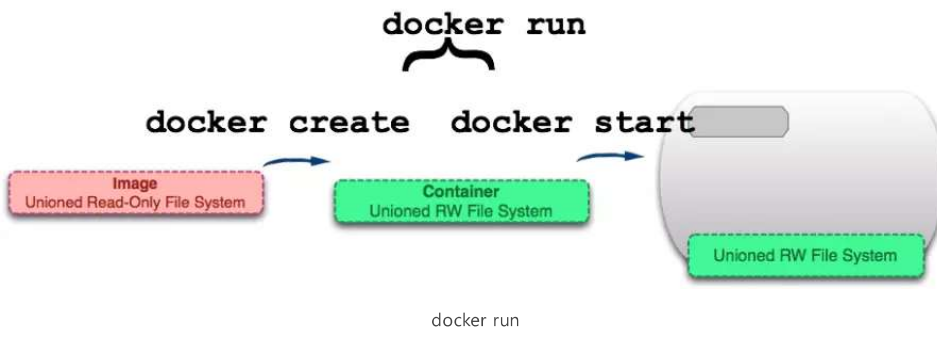
写下你的评论...

评论6 赞85 ...

```
46 # 运行一个带命令且在后台不断执行的容器，不直接展示容器内部信息。
47 $ docker run -d ubuntu:latest ping www.docker.com
48
49 # 运行一个在后台不断执行的容器，同时带有命令，程序被终止后还能重启继续跑，可用控制台管理。
50 $ docker run -d --restart=always ubuntu:latest ping www.docker.com
51
52 # 为容器指定一个名字
53 $ docker run -d --name=ubuntu_server ubuntu:latest
54
55 # 容器暴露80端口并指定宿主机81端口与其通信（宿主机端口:容器端口）
56 $ docker run -d --name=ubuntu -p 81:80 ubuntu:latest
57
58 # 指定容器内目录与宿主机目录共享（宿主机目录:容器目录）
59 $ docker run -d --name=ubuntu_server -v /home/www:/var/www ubuntu:latest
60
# 设置宿主机与docker的共享目录
$ docker run -d -i -t -p 80:80 -v /share/swoft:/var/www/swoft --name swoft swoft/swoft /b
```

使用 `docker run` 启动容器，docker在后台的标准操作流程：

1. 检查本地是否存在指定的镜像，若不存在则从公有仓库下载。
2. 使用镜像创建并启动容器
3. 分配一个文件系统，并在只读的镜像层外面挂载一层可读可写层。
4. 从宿主机配置的网桥接口中桥接一个虚拟机接口到容器中去
5. 从地址池分配一个IP地址给容器
6. 执行用户指定的应用程序
7. 执行完毕后容器被终止



docker create

```
1 # create 使用指定的镜像创建容器，与run命令不同的是，使用create命令只能创建容器而并不启动。
2 $ docker create [选项] <镜像名称,id> <命令> [参数]
3
4 # 选项
5 -a, --attach=[] 将标准输入、标准输出、标准错误链接到容器
6 -C, --cpu-shares=0 设置CPU资源分配，默认为1024。
7 -e, --env=[] 向容器设备环境变量，用于传递设置或密码。
8 -h, --hostname="" 设置容器主机名
9 -i, --interactive=false 激活标准输入，即使未与容器连接，也维持标准输入。
10 -m, --memory="" 设置内存限制，格式<数字><单位>，单位可用b、k、m、g。
11 -P, --publish-all=false 将连接到主机的容器的所有端口暴露在外
12 -p, --publish=[] 将连接到主机的容器的特定端口暴露在外，主要用于暴露web服务器的端口。
13 -t, --tty=false 使用TTY模式，若要使用Bash则必须设置该选项。
14 -u, --user="" 设置容器运行时要使用的Linux用户账户和UID
15 -v, --volume=[] 设置数据卷，设置要与主机共享的目录，不将文件保存到容器而是直接保存到主机，在主机目录后添
16 -w, --workdir="" 设置容器内部要运行进程的目录
17 --add-host=[] 向容器的/etc/hosts添加主机名与IP地址
18 --cap-add=[] 设置容器中使用的cgroups的特定capability，若设置为all则使用所有的capability。
19 --cap-drop=[] 从容器中删除cgroup的特定capability。
20 --cidfile="" 设置cid文件路径，cid中存储着所创建容器的id。
21 --cpuset="" 在多核CPU中设置要运行容器的核心数
```

```
29 | --expose=[] 仅连接容器的端口和主机，并不暴露在外。
30 | --link=[] 进行容器连接，格式 <容器名称>:<别名>
31 | --lxc-conf=[] 若使用LXC驱动则可设置LXC选项
32 | --name 设置容器名称
33 | --net="bridge" 设置容器的网络模式
34 | --privileged=false 在容器内部使用主机的所有linux内核功能
35 | --restart="" 设置容器内部进程终止时重启策略
36 | --security-opt=[] 设置SELinux、AppArmor选项
37 | --volumn-from=[] 连接数据卷容器，设置格式未<容器名称,id>:<:ro, :rw>，默认读写设置遵从-v选项的设置。
38 |
39 | # 使用docker镜像nginx:latest创建一个容器名为ubuntu_serve
    $ docker create --name ubuntu_serve ubuntu:latest
    $ docker create -it --name ubuntu_server ubutnu:latest /bin/bash
```

广告

docker start

```
1 | # start用于启动容器
2 | $ docker start <选项><容器名称, id>
3 |
4 | # 选项
5 | -a, --attach=false 将标准输入、标准输出、标准错误连接到容器，传递所有信号。
6 | -i, --interactive=false 激活标准输入
```

docker stop

```
1 | # stop用于终止容器
2 | $ docker stop <选项><容器名称, id>
3 |
4 | # 选项
5 | -t, --timeout=10 设置终止容器前的等待时间，单位为秒。
```

docker restart

```
1 | # restart 用于重启容器
2 | $ docker restart [选项] <容器名称, id>
```

docker pause

```
1 | # pause 用于暂停容器中所有的进程
2 | $ docker pause [选项] <容器名称, id>
```

docker unpause

```
1 | # unpause用于重启使用pause命令暂停的容器
2 | $ docker unpause <容器名称, id>
```

docker kill

```
1 | # kill用于杀掉一个运行中的容器，发送SIGKILL信号来停止的主进程。
2 | $ docker kill [选项] <容器名称, id>
3 |
4 | # 选项
5 | -s 向容器发送一个信号
6 |
7 | # 杀死运行中的容器nginx
8 | $ docker kill -s KILL nginx
```

docker rm

写下你的评论...

 评论6  赞85 ...

```
3 |  
4 | # 选项  
5 | -f 通过SIGKILL信号强制删除一个运行中的容器  
6 | -l 移除容器间的网路连接而非容器本身  
7 | -v 删除与容器关联的卷  
8 |  
9 | # 强制删除容器test  
10 | $ docker rm -f test  
11 |  
12 | # 删除容器test并删除挂载的数据卷  
13 | $ docker rm -v test
```

docker exec

```
1 | # exec 用于在运行中的容器中执行命令  
2 | $ docker exec [选项] <容器名称, id> <命令> [参数]  
3 |  
4 | # 选项  
5 | -d 分离模式即在后台运行  
6 | -i 即使没有附加也保持STDIN标准输入打开  
7 | -t 分配一个伪终端  
8 |  
9 | # 在容器test中以交互模式执行容器内/root/test.sh脚本  
10 | $ docker exec -it test /bin/sh /root/test.sh
```

容器操作

docker ps

```
1 | # ps用于输出容器列表  
2 | $ docker ps <选项>  
3 |  
4 | # 选项  
5 | -a, --all=false 列出所有容器，不带-a则输出当前正在运行的容器。  
6 | --before="" 列出特定容器创建前的容器，包含停止的容器。  
7 | -f, --filter=[] 设置输出过滤  
8 | -l, --latest=false 列出最后创建的容器包含停止的容器  
9 | -q, --quiet=false 只输出容器的ID  
10 |  
11 | $ docker ps  
12 | CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
```

docker top

```
1 | # top用户显示容器中正在 运行的进程信息  
2 | $ docker top <容器名称, id> <ps选项>  
3 |  
4 | $ docker top
```

docker attach

```
1 | # attach 用于将标准输入与标准输出连接到正在运行的容器  
2 | $ docker attach <选项><容器名称, id>
```

docker logs

```
1 | # logs用于获取容器的日志  
2 | $ docker logs [选项] <容器名称,id>  
3 |  
4 | # 选项  
5 | -f 跟随日志输出
```

```
11 | $ docker logs -f test
12 |
13 | # 查看test容器从2019年4月1日以后的最新10条日志
14 | $ docker logs --since="2019-04-01" --tail=10 test
```

如何动态跟踪并查看myswoft容器的日志详情呢？

```
1 | docker@ ~$: docker logs -tf --details myswoft
```

广告

如何清理日志文件呢？

Docker日志默认存储位于：`/var/lib/docker/containers/<container_id>/*.log`

```
1 | # 在docker中切换为root用户
2 | docker@default: ~$ su -i
3 |
4 | # 查看docker容器的id
5 | root@default: ~$ docker ps -a | grep myswoft
6 | d26ff0ff1bc0          swoft/swoft          "php /var/www/swoft/..."  2 weeks ago          Up 22
7 |
8 | # 查看容器日志文件大小
9 | root@default: ~$ ls -lh $(find /var/lib/docker/containers/ -name *-json.log)
10 | -rw-r-----   1 root    root              0 May 14 13:32 /var/lib/docker/containers/d26ff0ff1
11 |
12 | # 进入日志目录
13 | root@default: ~$ cd /var/lib/docker/containers/d26ff0ff1bc0dc0221ce5be7f78da96123a0ddb379a
14 |
15 | # 临时清理日志文件
16 | root@default: /var.../$ truncate -s 0 /var/lib/docker/containers/<container_id>/*-json.log
17 |
18 | # 临时清理日志文件
19 | root@default: /var.../$ cat /dev/null > *-json.log
20 |
21 |
22 | # 使用rm删除日志文件后是需要重启容器的，如果容器运行状态下Linux进程会引用着不会释放磁盘空间。
    root@default /var.../$ rm -rf *-json.log
```

docker port

```
1 | # port用于列出指定容器的端口映射，或者用于查找将PRIVATE_PORT NAT到面向公众的端口。
2 | $ docker port [选项] <容器名称,id> [PRIVATE_PORT[/PROTO]]
3 |
4 | # 查看test容器的端口映射情况
5 | $ docker port test
```

镜像仓库

docker search

```
1 | # search用于在docker hub中搜索镜像
2 | $ docker search <选项><搜索关键词>
3 |
4 | # 选项
5 | --automated=false 只显示由docker hub的automated build创建的镜像
6 | --no-trunc=false 显示所有因内容过长而省略的部分
7 | -s, --stars=0 显示有特定星级以上的镜像
```

本地镜像

docker images

写下你的评论...

评论6 赞85 ...

```
6 | -f, --filter=[] 设置输出结果过滤，若设置为dangling=true则仅输出无名镜像。
7 | --no-trunc=false 显示所有因内容过长而省略的部分
8 |
9 | $ docker images
10 | REPOSITORY          TAG                 IMAGE ID           CREATED            SIZE
```

docker rmi

广告

```
1 | # rmi 用于删除镜像，若没有指定标签则会删除latest标签。
2 | $ docker rmi <注册名称>/<镜像名称, id>:<标签>
3 |
4 | # 选项
5 | -f, --force=false 强制删除镜像
6 | --no-prune=false 不删除不带标签的父级镜像
7 |
8 | # 删除所有镜像
9 | $ docker rmi $(docker images -aq)
```

docker tag

```
1 | # tag用于设置镜像标签
2 | $ docker tag <选项><镜像名称>:<标签><注册地址, 用户名>/<镜像名称>:<标签>
3 |
4 | # 选项
5 | -f, --force=false 强制设置，即使已拥有标签，如远程仓库设置标签。
```

docker save

```
1 | # save用于将镜像保存为tar包文件
2 | $ docker save <选项><镜像名称>:<标签>
3 |
4 | # 选项
5 | -o, --output="" 设置保存时的文件名称
```

若不设置 `-o` 选项，`tar` 包文件会输出到标准输出，所以必须设置重定向。如果仅指定镜像名称但没有指定标签，则会将所有标签都保存到一个 `tar` 包文件中。

信息查看

docker info

```
1 | # info用于显示当前系统信息、docker容器和镜像数量、设置等信息。
2 | $ docker info
3 | Containers: 0
4 |   Running: 0
5 |   Paused: 0
6 |   Stopped: 0
7 | Images: 0
8 | Server Version: 18.09.5
9 | Storage Driver: overlay2
10 |   Backing Filesystem: extfs
11 |   Supports d_type: true
12 |   Native Overlay Diff: true
13 | Logging Driver: json-file
14 | Cgroup Driver: cgroupfs
15 | Plugins:
16 |   Volume: local
17 |   Network: bridge host macvlan null overlay
18 |   Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
19 | Swarm: inactive
20 | Runtimes: runc
21 | Default Runtime: runc
22 | --
```

写下你的评论...

 评论6  赞85 ...



```
29 | Profile: default
30 | Kernel Version: 4.14.111-boot2docker
31 | Operating System: Boot2Docker 18.09.5 (TCL 8.2.1)
32 | OSType: linux
33 | Architecture: x86_64
34 | CPUs: 1
35 | Total Memory: 989.4MiB
36 | Name: default
37 | ID: 7ST2:CIQM:GLVF:AUF2:QFKR:N2LB:FS07:V6UJ:5IFN:MQVZ:WK7L:TGSS
38 | Docker Root Dir: /mnt/sda1/var/lib/docker
39 | Debug Mode (client): false
40 | Debug Mode (server): false
41 | Registry: https://index.docker.io/v1/
42 | Labels:
43 |   provider=virtualbox
44 | Experimental: false
45 | Insecure Registries:
46 |   127.0.0.0/8
   | Live Restore Enabled: false
```

广告

docker version

```
1 | # version用户输出docker的版本信息
2 | $ docker version
3 | Client:
4 |   Version:      18.03.0-ce
5 |   API version:  1.37
6 |   Go version:   go1.9.4
7 |   Git commit:   0520e24302
8 |   Built: Fri Mar 23 08:31:36 2018
9 |   OS/Arch:      windows/amd64
10 |  Experimental:  false
11 |  Orchestrator:  swarm
12 |
13 | Server: Docker Engine - Community
14 |  Engine:
15 |    Version:      18.09.5
16 |    API version:  1.39 (minimum version 1.12)
17 |    Go version:   go1.10.8
18 |    Git commit:   e8ff056dbc
19 |    Built:        Thu Apr 11 04:50:00 2019
20 |    OS/Arch:      linux/amd64
21 |    Experimental: false
```

未完待续...



85人点赞 >



1人踩 >



docker



JunChow520
拥有857钻 (约146.92元)

关注

"小礼物走一走，来简书关注我"

赞赏

广告



写下你的评论...



写下你的评论...



评论6



赞85



Docker命令



JunChow520

关注

赞赏支持

似乎95%的命令都熟悉了

👍 赞 💬 回复



前端无聊
07.13 14:32

@Pyanko 还是动手能让我记得久，记得稳

💬 回复

广告

📝 添加新评论



七星小鱼
4楼 05.28 02:19

写得挺好，一起努力！

👍 赞 💬 回复



zheng10072
3楼 05.07 17:13

很全，很强

👍 赞 💬 回复



夕雅y
2楼 05.04 15:27

👤❤️

👍 赞 💬 回复

被以下专题收入，发现更多相似内容



.Net Core



容器



Docker



工具



程序猿的进阶屋



linux&g...



OpsDev

展开更多 ▾



写下你的评论...



评论6



赞85

