Docker 比较好的学习资料

<https://yeasy.gitbooks.io/docker_practice/>

Dockerfile最佳实践， 描写dockerfile编写的相关知识

<https://yeasy.gitbooks.io/docker_practice/appendix/best_practices.html>

sudo yum -y install epel-release

sudo yum install -y git nodejs docker python-pip mysql mysql-devel mysql-utilities gcc+ gcc-c++

sudo pip install --upgrade pip

sudo pip install docker-compose

添加docker镜像加速器

sudo mkdir -p /etc/docker

sudo tee /etc/docker/daemon.json <<-'EOF'

{

"registry-mirrors": ["http://0ea02bcd.m.daocloud.io"]

}

EOF

启动docker守护进程

sudo systemctl daemon-reload

sudo systemctl restart docker

1、构建docker 镜像实例 注意最后的路径.

|  |
| --- |
| docker build -t myimage . |

2、启动镜像

|  |
| --- |
| sudo docker run -d -p 9990:8080 test |

3、查看所有docker容器

|  |
| --- |
| docker ps -a |

4、删除docker容器

|  |
| --- |
| docker rm container\_id |

5、查询所有docker 镜像

|  |
| --- |
| docker images |

6、删除指定docker 镜像

|  |
| --- |
| docker rmi image\_id |

7、根据容器id进入容器里面

|  |
| --- |
| sudo docker exec -it 容器id bash |

退出容器到宿主机，用exit

8、查看容器详细信息

|  |
| --- |
| sudo docker inspect 容器id |

9、停掉容器

|  |
| --- |
| sudo docker stop 容器id |

10、运行docker容器

|  |
| --- |
| sudo docker run -d -p 9092:8080 --name yubo-yuce-callshell-container --env category\_shell\_command="ssh -p 58958 userprofile@169.44.23.158 'sh /home/userprofile/yonggan/user-group-task.sh " -v /home/boris/.ssh/id\_rsa.pub:/root/.ssh/id\_rsa.pub -v /home/boris/.ssh/id\_rsa:/root/.ssh/id\_rsa yubo-yuce-callshell-image |

10、搜索远程仓库中的images

|  |
| --- |
| sudo docker search 镜像名 |

11、列出本地images

|  |
| --- |
| sudo docker images |

12、从远程镜像仓库中拉去image

|  |
| --- |
| sudo docker pull image |

13、在容器中安装程序

|  |
| --- |
| sudo docker run image 命令(apt-get intall -y ping) |

14、将容器保存为镜像

|  |
| --- |
| sudo docker commit 容器id 镜像名 |

执行命令后会返回 镜像的id 类似：

sha256:c0c05029df0a5b97a7760bade1fcda9bb8a20311eecbfa8215703b85efda5e70

15、查看运行中的容器

|  |
| --- |
| sudo docker inspect 容器id |

16、登录dockerHub

|  |
| --- |
| sudo docker login |

然后输入用户名密码

 如图：表示登录成功。

17、将镜像push到远程仓库。

|  |
| --- |
| sudo docker push 用户名/镜像名 |

18、查看docker控制台日志

|  |
| --- |
| sudo docker logs -f \*\*\*(容器名） |

Docker服务yum安装，在centos7上

<https://docs.docker.com/install/linux/docker-ce/centos/>

docker 服务器开机自启动：

1.systemctl is-enabled docker.service  检查服务是否开机启动

2.systemctl enable docker.service  将服务配置成开机启动

3.systemctl start docker.service  启动服务

systemctl  相关其他命令：

systemctl disable docker.service 禁止开机启动

systemctl stop docker.service  停止  
systemctl restart docker.service  重启

//------------------------------------------------------

容器开机启动：

创建容器时候指定restart参数：

docker run    -it -p 6379:6379 --restart=always  --name redis -d redis

对已经创建的容器用docker update 更新:

docker update --restart=always  xxx

--restart具体参数值详细信息 :

no - 容器退出时，不重启容器  
on-failure - 只有在非0状态退出时才从新启动容器  
always - 无论退出状态是如何，都重启容器

sudo docker logs -f 容器名

sudo docker logs -f -t --tail 行数 容器名

查看容器挂载信息

docker inspect 28b941608555 | grep Mounts -A 1000

docker inspect 28b941608555 查看全部信息

获取容器ip地址

docker inspect -f {{.NetworkSettings.IPAddress}} 28b941608555

docker环境执行docker exec 容器名 容器内命令的方式也可以获取相关信息，举例：

1. 获取容器的hostname：docker exec tomcat001 hostname
2. 获取容器ip地址：docker exec tomcat001 ip addr
3. 获取容器环境变量：docker exec tomcat001 env

Docker run 的一些参数 –d –user –p –v –e 等

docker run -d --user $ID --volume "$PWD/data:/var/lib/grafana" -p 3000:3000 grafana/grafana:5.1.0

docker run -d --user $ID --name grafana-v4 -p 3000:3000 -v /home/soft/grafana/grafana.ini:/etc/grafana/grafana.ini -v /home/soft/grafana/grafana/:/var/lib/grafana/ -e GF\_SMTP\_ENABLED=true -e GF\_SMTP\_SKIP\_VERIFY=true -e GF\_SMTP\_HOST=smtp.mxhichina.com:25 -e GF\_SMTP\_USER=yuboa@yingyinglicai.com -e GF\_SMTP\_PASSWORD=yylc123@ -e GF\_SMTP\_FROM\_ADDRESS=yuboa@yingyinglicai.com grafana/grafana

docker run \

-d --name grafana -p 3000:3000 \

-e "GF\_SERVER\_ROOT\_URL=http://grafana.server.name" \

-e "GF\_SECURITY\_ADMIN\_PASSWORD=newpwd" \

--user $ID --volume "$PWD/data:/var/lib/grafana" \

grafana/grafana

CMD 和ENTRYPOINT差异

场景一：让镜像变成像命令一样使用

FROM centos

CMD [ "curl", "-s", "https://ip.cn" ]

cmd命令 不能 docker run image 后面再加 -i等参数 他会作为可执行命令覆盖 Dockerfile中的 CMD 命令

FROM centos

ENTRYPOINT [ "curl", "-s", "https://ip.cn" ]

entrypoint 能在 docker run image 后面再加 -i等其他参数。新增的部分 作为entrypoint的一部分来使用。

entrypoint中的数组，不能写一起，"curl -s https://ip.cn" 这样是不行的。