docker 详解

1、 安装 docker

安装必要的一些系统工具
yum install -y yum-utils device-mapper-persistent-data lvm2
添加软件源信息
yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
sed -i 's+download.docker.com+mirrors.aliyun.com/docker-ce+' /etc/yum.repos.d/docker-ce.repo
更新并安装 Docker-CE
yum -y install docker-ce
开启 Docker 服务
systemctl start docker

2、 配置镜像加速(一定配置)

```
# mkdir -p /etc/docker
# vim /etc/docker/daemon.json   //添加以下三行
{
"registry-mirrors": ["https://wli421u6.mirror.aliyuncs.com"]
}

# systemctl daemon-reload
# systemctl restart docker
```

3、 镜像分类

• BusyBox：一个极简版的 Linux 系统，集成了 100 多种常用 Linux 命令，大小不到 2MB，被称为“Linux 系统的瑞士军刀”，适用于简单测试场景；

• Alpine：一个面向安全的轻型 Linux 发行版系统，比 BusyBox 功能更完善，大小不到 5MB，是官网推荐的基础镜像，由于其包含了足够的基础功能和体积较小，在生产环境中最常用；

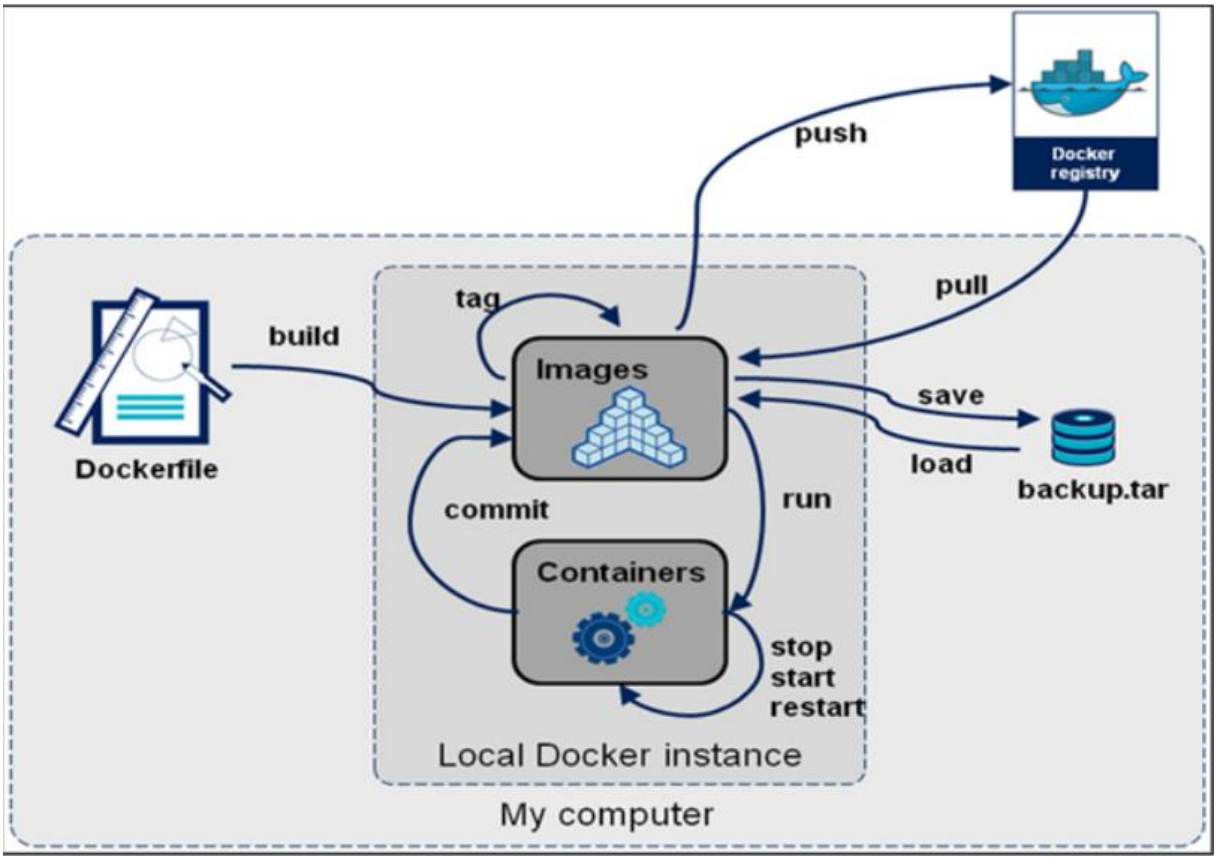
• Debian/Ubuntu： Debian 系列操作系统，功能完善，大小约 170MB，适合研发环境；

• CentOS/Fedora：都是基于 Redhat 的 Linux 发行版，企业级服务器常用操作系统，稳定性高，大小约 200MB，适合生产环境使用。

镜像分类：

1、基础镜像---　Centos
2、服务镜像---　nginx　tomcat　apache 之类
3、项目镜像---- 上线项目

4、 docker 的基本使用



images 镜像管理

查看本机的 image
docker image ls
docker images

获取 image
docker search centos　　#　此命令会在 docker 官方仓库查找所需镜像,速度较慢.
//有路径的镜像一般为私人上传的镜像，无路径的一般为官方镜像
docker pull centos　　#　从默认仓库拉取 centos 镜像最新版

删除 image
docker image　rm　镜像　　#镜像指　镜像 ID　或　镜像名称:tag
docker rmi　镜像

查看 image 的详细信息
docker inspect　[选项]　镜像

Containers 容器管理

查看容器

```
# docker ps                    # 查看正在运行的容器
# docker ps   -a               # 查看所有容器
```

启动容器

```
# docker run -itd   -h centos.docker.qf.com --name centos centos
```

常用选项：

```
    -i                   # 允许你对容器内的标准输入 (STDIN) 进行交互
    -t                   # 在新容器内指定一个伪终端或终端
    -d                   # 在后台运行容器,返回容器 ID
    -h                   # 指定容器的主机名,如果不指定,会随机生成一个
    --rm                 # 容器停止后删除掉,默认不会删除
    --name               # 指定容器的名称
    --network            # 指定容器的网络连接方式,默认为 NAT.
    --ip                 # 指定容器的 IP 地址
    -p <宿主端口>:<容器端口>    # 端口映射,将容器指定端口映射到宿主机的指定端口. 可以用多个-p 选项指定多个端口映射
    -p 80                # 将容器的 80 端口映射到宿主机的随机端口   1-65535
    -P                   # 将容器的所有端口映射到宿主机的随机端口
    -v <宿主目录>:<容器目录>    #将宿主机的指定目录映射到容器的指定目录
    --privileged=ture                                  #需要修改某些特定的参数需要加上此选项,正常运行一个容器不建议开放这个权限，需要在容器内部使用 systemctl 命令需加上，对应环境改为/usr/sbin/init
```

其他容器操作命令

```
# docker create [选项] 镜像          # 创建一个容器但不运行,选项基本和 run 命令相同
# docker ps [-a]                     # 查看运行的容器 [-a 所有容器]
# docker top 容器                    # 查看容器的进程信息
# docker stop|start|restart|kill 容器     # 启停容器
# docker pause|unpause 容器          # 暂停|恢复容器
# docker rm 容器                     # 删除停止的容器
# docker rm -f 容器                  # 强制删除运行的容器
# docker rm `docker ps -a -q`        # 删除所有容器
# docker logs 容器                   # 查看容器日志
# docker cp   container:src_file   dst_path    # 将容器中的文件复制到宿主机上
# docker cp src_file   container:dst_path      # 将宿主机上的文件复制到容器中
# docker inspect   [选项]   容器              # 以 json 格式显示出容器的具体信息
```

连接容器

对于正在运行的容器,我们可以在宿主机上连接容器

```
# docker exec -it centos /bin/bash        # 可用 exit 命令退出,不影响容器
# docker attach centos                    # 通过 attach 连接容器,使用 exit 退出后容器会关闭,
```

当多个窗口同时使用该命令进入该容器时，
所有的窗口都会同步显示．如果有一个窗口阻塞了，那么其他窗口也无法再进行操作
若不想退出后停止容器,可通过快捷键 ctrl+pq 退出

容器 ---> 镜像
对容器所作的修改保存在容器中，一旦容器被删除了，修改也没有了.
为了永久保存，可以将容器打包成镜像:
docker commit -m "描述信息"　　容器　　镜像名[:tag]

基于镜像
导出:
[root@docker ~]# docker save centos:latest > /bak/docker-centos_latest.bak

导入:
docker load < /bak/docker-centos_latest.bak

基于容器
导出:
docker export　centos7 > /bak/docker-centos7.bak

导入:
docker import /bak/centos7.bak　centos7　　#导入了镜像

5、 部署 Harbor-私有仓库

配置过程
1. 安装 docker-compose
Docker Compose 是 docker 提供的一个命令行工具，用来定义和运行由多个容器组成的应用。
使用 compose，我们可以通过 YAML 文件声明式的定义应用程序的各个服务，并由单个命令完成应用的创建和启动。
yum install -y　docker-compose

2、下载并且解压

2. 下载 harbor
wget https://storage.googleapis.com/harbor-releases/release-1.9.0/harbor-offline-installer-v1.9.1.tgz
tar xf harbor-offline-installer-v1.9.1.tgz　-C /usr/local

3、安装

cd /usr/local/harbor/
vim harbor.yml　　//由于没有域名，修改为 IP 直接访问

./prepare

```
Generated configuration file: /config/log/logrotate.conf
Generated configuration file: /config/log/rsyslog_docker.conf
Generated configuration file: /config/nginx/nginx.conf
Generated configuration file: /config/core/env
Generated configuration file: /config/core/app.conf
Generated configuration file: /config/registry/config.yml
Generated configuration file: /config/registryctl/env
Generated configuration file: /config/db/env
Generated configuration file: /config/jobservice/env
Generated configuration file: /config/jobservice/config.yml
Generated and saved secret to file: /secret/keys/secretkey
Generated certificate, key file: /secret/core/private_key.pem, cert file: /secret/registry/root.crt
Generated configuration file: /compose_location/docker-compose.yml
```

./install.sh

```
✔ ----Harbor has been installed and started successfully.----

Now you should be able to visit the admin portal at http://192.168.10.11.
For more details, please visit https://github.com/goharbor/harbor .
```

6、安装完成后进入 web 界面

用户名: admin

密码: 见配置文件　默认是： Harbor12345

7、创建项目

先登录 harbor 的 Web 界面，创建一个项目

这个项目就相当于公司中不同的项目组，每个项目组分别管理各自的项目镜像，后期该项目不需要时，可直接删除该项目。

案例：上传 busybox 镜像到 Harbor 私有仓库 test 项目

docker pull busybox

docker login 124.71.170.184

admin：admin

password：Harbor12345

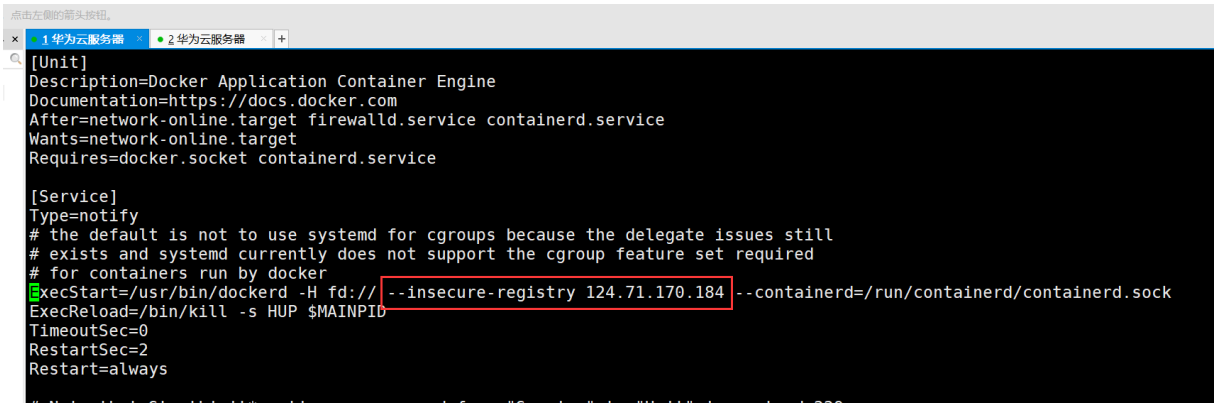
```
[root@QF harbor]# docker login 124.71.170.84
Username: admin
Password:
Error response from daemon: Get "https://124.71.170.84/v2/": dial tcp 124.71.170.84:443: connect: connection refused
[root@QF harbor]#
```

https的加密错误

解决该错误：

vim /usr/lib/systemd/system/docker.service

--insecure-registry 124.71.170.184　加上这个



systemctl daemon-reload

systemctl restart docker

docker tag busybox:latest 124.71.170.184/test/busybox:latest

docker push 124.71.170.184/test/busybox:latest