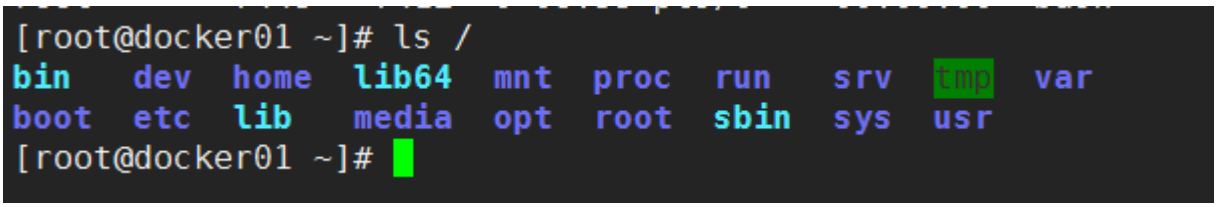
docker容器

1：什么是容器？

容器就是在隔离的环境运行的一个进程，如果进程停止，容器就会销毁。隔离的环境拥有自己的系统文件，ip地址，主机名等

kvm虚拟机，linux，系统文件



程序：代码，命令 进程：正在运行的程序

2：容器和虚拟化的区别

linux容器技术，容器虚拟化和kvm虚拟化的区别 kvm虚拟化： 需要硬件的支持，需要模拟硬件，可以运行不同的操作系统，启动时间分钟级(开机启动流程)

linux开机启动流程： bios开机硬件自检 根据bios设置的优先启动项boot 网卡 硬盘 u盘 光驱 读取mbr引导 2T UEFI(gpt分区) mbr硬盘分区信息，内核加载路径,
加载内核 启动第一个进程/sbin/init systemd 系统初始化完成 运行服务(nginx，httpd，mysql) 。。。

容器启动流程： 共用宿主机内核： 第一个进程直接启动服务(nginx，httpd，mysql)

容器：共用宿主机内核，轻量级，损耗少，启动快，性能高，只能运行linux系统 虚拟机：需要硬件的支持，需要模拟硬件，需要走开机启动流程，可以运行不同的操作系统

3:容器技术的发展过程：

1）:chroot技术，新建一个子系统（拥有自己完整的系统文件）

参考资料：https://www.ibm.com/developerworks/cn/linux/l-cn-chroot/ chang root

作业1：使用chroot监狱限制SSH用户访问指定目录和使用指定命令(cp,ls) https://linux.cn/article-8313-1.html ls

2）：linux容器(lxc) linux container(namespaces 命名空间 隔离环境 及 cgroups 资源限制)

cgroups 限制一个进程能够使用的资源。cpu，内存，硬盘io

kvm虚拟机：资源限制（1c 1G 20G）

##需要使用epel源 #安装epel源 yum install epel-release -y

#编译epel源配置文件 vi /etc/yum.repos.d/epel.repo [epel] name=Extra Packages for Enterprise Linux 7 - $basearch baseurl=https://mirrors.tuna.tsinghua.edu.cn/epel/7/$basearch #mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basearch failovermethod=priority enabled=1 gpgcheck=1 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

[epel-debuginfo] name=Extra Packages for Enterprise Linux 7 - $basearch - Debug baseurl=https://mirrors.tuna.tsinghua.edu.cn/epel/7/$basearch/debug #mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-debug-7&arch=$basearch failovermethod=priority enabled=0 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7 gpgcheck=1

[epel-source] name=Extra Packages for Enterprise Linux 7 - $basearch - Source baseurl=https://mirrors.tuna.tsinghua.edu.cn/epel/7/SRPMS #mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-source-7&arch=$basearch failovermethod=priority enabled=0 gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7 gpgcheck=1

##安装lxc yum install lxc-* -y yum install libcgroup* -y yum install bridge-utils.x86_64 -y

##桥接网卡 [root@controller ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0 echo 'TYPE=Ethernet BOOTPROTO=none NAME=eth0 DEVICE=eth0 ONBOOT=yes BRIDGE=virbr0' >/etc/sysconfig/network-scripts/ifcfg-eth0

[root@controller ~]# cat /etc/sysconfig/network-scripts/ifcfg-virbr0 echo 'TYPE=Bridge BOOTPROTO=static NAME=virbr0 DEVICE=virbr0 ONBOOT=yes IPADDR=10.0.0.11 NETMASK=255.255.255.0 GATEWAY=10.0.0.254 DNS1=180.76.76.76' >/etc/sysconfig/network-scripts/ifcfg-virbr0

##启动cgroup systemctl start cgconfig.service

##启动lxc systemctl start lxc.service

##创建lxc容器 方法1: lxc-create -t download -n centos6 -- --server mirrors.tuna.tsinghua.edu.cn/lxc-images -d centos -r 6 -a amd64 方法2： lxc-create -t centos -n test

#####为lxc容器设置root密码： [root@controller ~]# chroot /var/lib/lxc/test/rootfs passwd Changing password for user root. New password: BAD PASSWORD: it is too simplistic/systematic BAD PASSWORD: is too simple Retype new password: passwd: all authentication tokens updated successfully.

##为容器指定ip和网关 vi /var/lib/lxc/centos7/config lxc.network.name = eth0 lxc.network.ipv4 = 10.0.0.111/24 lxc.network.ipv4.gateway = 10.0.0.254

##启动容器 lxc-start -n centos7

3) :docker容器

centos7.4 2G 10.0.0.11 docker01 host解析 centos7.4 2G 10.0.0.12 docker02 host解析

Docker是通过进程虚拟化技术（namespaces及cgroups cpu、内存、磁盘io等）来提供容器的资源隔离与安全保障等。由于Docker通过操作系统层的虚拟化实现隔离，所以Docker容器在运行时，不需要类似虚拟机（VM）额外的操作系统开销，提高资源利用率。 namespace 资源隔离 cgroups 进程的资源限制 kvm 虚拟磁盘文件，资源隔离 kvm 资源限制，--cpus --memory

docker 初期把lxc二次开发，libcontainer

docker的主要目标是"Build,Ship and Run any App,Angwhere",构建，运输，处处运行 部署服务，环境问题

一次构建，处处运行

docker是一种软件的打包技术

4：docker的安装

10.0.0.11：修改主机名和host解析

rm -fr /etc/yum.repos.d/local.repo curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo wget -O /etc/yum.repos.d/docker-ce.repo https://mirrors.ustc.edu.cn/docker-ce/linux/centos/docker-ce.repo sed -i 's#download.docker.com#mirrors.tuna.tsinghua.edu.cn/docker-ce#g' /etc/yum.repos.d/docker-ce.repo yum install docker-ce -y

5:docker的主要组成部分

docker是传统的CS架构分为docker client和docker server,向mysql一样

命令：docker version [root@controller ~]# docker version Client: Version: 17.12.0-ce API version: 1.35 Go version: go1.9.2 Git commit: c97c6d6 Built: Wed Dec 27 20:10:14 2017 OS/Arch: linux/amd64

Server: Engine: Version: 17.12.0-ce API version: 1.35 (minimum version 1.12) Go version: go1.9.2 Git commit: c97c6d6 Built: Wed Dec 27 20:12:46 2017 OS/Arch: linux/amd64 Experimental: false

docker info（如果要做监控）

docker主要组件有：镜像、容器、仓库, 网络，存储

启动容器必须需要一个镜像，仓库中只存储镜像 容器---镜像---仓库

docker初次体验： 安装Nginx步骤： 官网下载Nginx源码包wget
tar 创建Nginx用户

编译安装 ./config.... 修改配置文件， 启动

6：启动第一个容器

##配置docker镜像加速 vi /etc/docker/daemon.json { "registry-mirrors": ["https://registry.docker-cn.com"] }

docker run -d -p 80:80 nginx run（创建并运行一个容器） -d 放在后台 -p 端口映射 nginx docker镜像的名字

7：docker的镜像管理

搜索镜像 docker search 选镜像的建议： 1，优先考虑官方 2，stars数量多

官方镜像仓库地址：hub.docker.com

获取镜像 docker pull（push）镜像加速器：阿里云加速器，daocloud加速器，中科大加速器，Docker 中国官方镜像加速：https://registry.docker-cn.com 官方pull docker pull centos:6.8（没有指定版本，默认会下载最新版）私有仓库pull docker pull daocloud.io/huangzhichong/alpine-cn:latest

##配置docker镜像加速 vi /etc/docker/daemon.json { "registry-mirrors": ["https://registry.docker-cn.com"] }

查看镜像列表 docker images or docker image ls 删除镜像 docker rmi 例子：docker image rm centos:latest 导出镜像 docker save 例子：docker image save centos > docker-centos7.4.tar.gz 导入镜像 docker load 例子：docker image load -i docker-centos7.4.tar.gz

8：docker的容器管理

docker run -d -p 80:80 nginx:latest
run（创建并运行一个容器） -d 放在后台 -p 端口映射 -v 源地址(宿主机):目标地址(容器)

nginx docker镜像的名字

docker run -it --name centos6 centos:6.9 /bin/bash -it 分配交互式的终端 --name 指定容器的名字 /bin/sh覆盖容器的初始命令

启动容器*** docker run image_name

```
docker run ==== docker create  + docker start
```

停止容器 docker stop CONTAINER_ID 杀死容器 docker kill container_name 查看容器列表 docker ps（-a -l -q）

进入容器(目的，调试，排错) *** docker exec （会分配一个新的终端tty） docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

```
docker exec -it  容器id或容器名字 /bin/bash（/bin/sh）
docker attach（使用同一个终端）
    docker attach [OPTIONS] CONTAINER
nsenter(安装yum install -y util-linux 弃用)
```

删除容器 docker rm 批量删除容器 docker rm -f `docker ps -a -q` 总结：docker容器内的第一个进程（初始命令）必须一直处于前台运行的状态（必须夯住），否则这个容器，就会处于退出状态！

业务在容器中运行：夯住，启动服务

9：docker容器的网络访问（端口映射）

docker0：172.17.0.1 jumpserver：172.17.0.2 nginx：172.17.0.3

指定映射(docker 会自动添加一条iptables规则来实现端口映射) -p hostPort:containerPort -p ip:hostPort:containerPort 多个容器都想使用80端口 -p ip::containerPort(随机端口) -p hostPort:containerPort:udp -p 10.0.0.100::53:udp 使用宿主机的10.0.0.100这个ip地址的随机端口的udp协议映射容器的udp53端口 -p 81:80 –p 443:443 可以指定多个-p

随机映射 docker run -P （随机端口）

通过iptables来实现的端口映射

10：docker的数据卷管理

/usr/share/nginx/html

-v /opt/xiaoniao:/usr/share/nginx/html

持久化 数据卷(文件或目录) -v 卷名:/data -v src（宿主机的目录）:dst（容器的目录） 数据卷容器 --volumes-from（跟某一个已经存在的容器挂载相同的卷） 基于nginx启动一个容器，监听80和81，访问80，出现nginx默认欢迎首页，访问81，出现小鸟。 -p 80:80 -p 81:81 -v xxx：xxx -v xxx：xxxx 基于nginx多端口的多站点。

11：手动将容器保存为镜像 docker commit 容器id或者容器的名字 新的镜像名字[:版本号可选]

1）：基于容器制作镜像 docker run -it centos:6.9 ###### yum install httpd yum install openssh-server /etc/init.d/sshd start

vi /init.sh #!/bin/bash /etc/init.d/httpd start /usr/sbin/sshd -D

chmod +x /init.sh

2）将容器提交为镜像 docker commit oldboy centos6-ssh-httpd:v1

3）测试镜像功能是否可用

手动制作的镜像，传输时间长 镜像初始命令

制作一个kodexplorer网盘docker镜像。nginx + php-fpm（httpd + php）

12：dockerfile自动构建docker镜像 类似ansible剧本，大小几kb 手动做镜像：大小几百M+

dockerfile 支持自定义容器的初始命令

dockerfile主要组成部分： 基础镜像信息 FROM centos:6.9 制作镜像操作指令 RUN yum install openssh-server -y 容器启动时执行指令 CMD ["/bin/bash"] dockerfile常用指令： FROM 这个镜像的妈妈是谁？（指定基础镜像） MAINTAINER 告诉别人，谁负责养它？（指定维护者信息，可以没有） LABLE 描述，标签

```
RUN 你想让它干啥（在命令前面加上RUN即可）
ADD 给它点创业资金（会自动解压tar） 制作docker基础的系统镜像
WORKDIR 我是cd,今天刚化了妆（设置当前工作目录）
VOLUME 给它一个存放行李的地方（设置卷，挂载主机目录）
EXPOSE 它要打开的门是啥（指定对外的端口）(-P 随机端口)
CMD 奔跑吧，兄弟！（指定容器启动后的要干的事情）（容易被替换）
```

dockerfile其他指令： COPY 复制文件（不会解压）rootfs.tar.gz ENV 环境变量 ENTRYPOINT 容器启动后执行的命令（无法被替换，启容器的时候指定的命令，会被当成参数）

参考其他的dockerfile 官方dockerfile或者时速云镜像广场

13：docker镜像的分层（kvm 链接克隆，写时复制的特性） 镜像分层的好处：复用,节省磁盘空间，相同的内容只需加载一份到内存。 修改dockerfile之后，再次构建速度快

14:.容器间的互联（--link 是单方向的！！！） docker run -d -p 80:80 nginx docker run -it --link quirky_brown:web01 qstack/centos-ssh /bin/bash ping web01 lb ---> nginx 172.17.0.4 --> db01 172.17.0.3 --> nfs01 172.17.0.2

使用docker运行zabbix-server docker run --name mysql-server -t \ -e MYSQL_DATABASE="zabbix" \ -e MYSQL_USER="zabbix" \ -e MYSQL_PASSWORD="zabbix_pwd" \ -e MYSQL_ROOT_PASSWORD="root_pwd" \ -d mysql:5.7 \ --character-set-server=utf8 --collation-server=utf8_bin

docker run --name zabbix-java-gateway -t \ -d zabbix/zabbix-java-gateway:latest

docker run --name zabbix-server-mysql -t \ -e DB_SERVER_HOST="mysql-server" \ -e MYSQL_DATABASE="zabbix" \ -e MYSQL_USER="zabbix" \ -e MYSQL_PASSWORD="zabbix_pwd" \ -e MYSQL_ROOT_PASSWORD="root_pwd" \ -e ZBX_JAVAGATEWAY="zabbix-java-gateway" \ --link mysql-server:mysql \ --link zabbix-java-gateway:zabbix-java-gateway \ -p 10051:10051 \ -d zabbix/zabbix-server-mysql:latest

docker run --name zabbix-web-nginx-mysql -t \ -e DB_SERVER_HOST="mysql-server" \ -e MYSQL_DATABASE="zabbix" \ -e MYSQL_USER="zabbix" \ -e MYSQL_PASSWORD="zabbix_pwd" \ -e MYSQL_ROOT_PASSWORD="root_pwd" \ --link mysql-server:mysql \ --link zabbix-server-mysql:zabbix-server \ -p 80:80 \ -d zabbix/zabbix-web-nginx-mysql:latest

监控报警：微信报警，alpine
yum 安装zabbix好使

16：docker registry（私有仓库） ##普通的registry docker run -d -p 5000:5000 --restart=always --name registry -v /opt/myregistry:/var/lib/registry registry

上传镜像到私有仓库： a:给镜像打标签 docker tag centos6-sshd:v3 10.0.0.11:5000/centos6-sshd:v3 b:上传镜像 docker push 10.0.0.11:5000/centos6-sshd:v3

docker run -d 10.0.0.11:5000/centos6-sshd:v3 如果遇到报错： The push refers to repository [10.0.0.11:5000/centos6.9_ssh] Get https://10.0.0.11:5000/v2/: http: server gave HTTP response to HTTPS client

解决方法： vim /etc/docker/daemon.json { "insecure-registries": ["10.0.0.11:5000"] } systemctl restart docker

##带basic认证的registry yum install httpd-tools -y mkdir /opt/registry-var/auth/ -p htpasswd -Bbn oldboy 123456 >> /opt/registry-var/auth/htpasswd

docker run -d -p 5000:5000 -v /opt/registry-var/auth/:/auth/ -v /opt/myregistry:/var/lib/registry -e "REGISTRY_AUTH=htpasswd" -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd" registry

17:docker-compose(单机版的容器编排工具) ansible剧本

yum install -y python2-pip（需要epel源） pip install docker-compose（默认pypi源在国外） ##pip 加速

##详细指令 http://www.jianshu.com/p/2217cfed29d7

cd my_wordpress/ vi docker-compose.yml version: '3'

services: db: image: mysql:5.7 volumes: - db_data:/var/lib/mysql restart: always environment: MYSQL_ROOT_PASSWORD: somewordpress MYSQL_DATABASE: wordpress MYSQL_USER: wordpress MYSQL_PASSWORD: wordpress

wordpress: depends_on: - db image: wordpress:latest volumes: - web_data:/var/www/html ports: - "80:80" restart: always environment: WORDPRESS_DB_HOST: db:3306 WORDPRESS_DB_USER: wordpress WORDPRESS_DB_PASSWORD: wordpress volumes: db_data: web_data:

#启动 docker-compose up #后台启动 docker-compose up -d

18：重启docker服务，容器全部退出的解决办法 方法一：docker run --restart=always

方法二："live-restore": true docker server配置文件/etc/docker/daemon.json参考 { "registry-mirrors": ["http://b7a9017d.m.daocloud.io"], "insecure-registries":["10.0.0.11:5000"], "live-restore": true }

19：Docker Machine安装docker服务 Docker Machine 二进制 10.0.0.11 10.0.0.12 免密码登陆 从docker的官网下载二进制的包，去安装docker 10.0.0.13 免密码登陆

ansible： shell

20：Docker网络类型 None：不为容器配置任何网络功能，--net=none Container：与另一个运行中的容器共享Network Namespace，--net=container:containerID（K8S） Host：与宿主机共享Network Namespace，--net=host Bridge：Docker设计的NAT网络模型

21：Docker跨主机容器之间的通信macvlan

默认一个物理网卡，只有一个物理地址，虚拟多个mac地址

##创建macvlan网络 docker network create --driver macvlan --subnet 10.0.0.0/24 --gateway 10.0.0.254 -o parent=eth0 macvlan_1 ##设置eth0的网卡为混杂模式 ip link set eth1 promisc on ##创建使用macvlan网络的容器 docker run -it --network macvlan_1 --ip=10.0.0.200 busybox

作业1：用PIPEWORK为docker容器配置独立IP 作业2：docker跨主机容器间的通信flannel

22：Dcoker跨主机容器通信之overlay http://www.cnblogs.com/CloudMan6/p/7270551.html 1）准备工作 docker01上： docker run -d -p 8500:8500 -h consul --name consul progrium/consul -server -bootstrap 设置容器的主机名

consul：kv类型的存储数据库（key:value） docker01、02上： vim /etc/docker/daemon.json { "hosts": ["tcp://0.0.0.0:2376","unix:///var/run/docker.sock"], "cluster-store": "consul://10.0.0.13:8500", "cluster-advertise": "10.0.0.11:2376" }

vim /etc/docker/daemon.json vim /usr/lib/systemd/system/docker.service systemctl daemon-reload systemctl restart docker

2）创建overlay网络 docker network create -d overlay --subnet 172.16.1.0/24 --gateway 172.16.1.254 ol1

3）启动容器测试 docker run -it --network ol1 --name oldboy01 busybox /bin/bash 每个容器有两块网卡,eth0实现容器间的通讯,eth1实现容器访问外网

23：docker企业级镜像仓库harbor(vmware 中国团队) 第一步：安装docker和docker-compose

第二步：下载harbor-offline-installer-v1.3.0.tgz

第三步：上传到/opt,并解压

第四步：修改harbor.cfg配置文件 hostname = 10.0.0.11 harbor_admin_password = 123456

第五步：执行install.sh

###k8s的安装方法 kubernetes 二进制安装 安装最新版,步骤繁琐!! https://github.com/minminmsn/k8s1.13/blob/master/kubernetes/kubernetes1.13.1%2Betcd3.3.10%2Bflanneld0.10%E9%9B%86%E7%BE%A4%E9%83%A8%E7%BD%B2.md

kubeadm 安装(网络原因) https://www.qstack.com.cn/archives/425.html

minikube 安装(网络原因)

yum 安装(最容易 1.5)

go编译安装(大神级别)

k8s-master 管理者 kubelet --docker 启动容器 kubelet --docker

####制作一个只支持sshd服务的镜像 1):启动一个容器，并修改 docker run -it -p 1022:22 centos:6.8 /bin/bash yum install openssh-server -y echo 'root:123456'|chpasswd /etc/init.d/sshd start 测试：ssh远程登录

2）：将修改后的容器，保存为镜像 docker commit friendly_swartz centos6-ssh

3）测试新镜像，sshd是否可用 docker run -d -p 1122:22 centos6-ssh:latest /usr/sbin/sshd -D ssh root@10.0.0.11 -p 1122

####制作了一个支持sshd和httpd双服务的镜像 1）：启动一个容器，并修改 docker run -d -p 1122:22 centos6-ssh:latest /usr/sbin/sshd -D yum install httpd -y /etc/init.d/httpd start

vi /init.sh #!/bin/bash /etc/init.d/httpd start /usr/sbin/sshd -D

chmod +x /init.sh 2）：将修改后的容器，保存为镜像 docker commit 11bf5984784a centos6-httpd

3）测试新镜像，检测sshd和httpd是否可用 docker run -d -p 1222:22 -p 80:80 centos6-httpd:latest /init.sh

作业：使用Dockerfile完成kodexplorer网盘项目

课前回顾： Linux容器是与系统其他部分隔离开的一系列进程，从另一个系统镜像rootfs运行，并由该镜像提供支持进程所需的全部文件。 容器镜像包含了应用的所有依赖项，因而在从开发到测试再到生产的整个过程中，它都具有可移植性和一致性。

1：chroot，ldd 2：lxc namespace(6大命名空间)和cgroup 3：docker和kvm区别，docker镜像基础操作，docker容器日常操作，commit制作镜像，dockfile来自动构建镜像

####docker私有仓库registry 1）启动registry容器 docker run -d -p 5000:5000 --restart=always --name registry -v /opt/myregistry:/var/lib/registry registry

2）修改/etc/docker/daemon.json配置文件 { "registry-mirrors": ["https://registry.docker-cn.com"], "insecure-registries": ["10.0.0.11:5000"] }

3）重启docker服务 systemctl restart docker

4）为镜像打标签 docker tag centos:6.8 10.0.0.11:5000/oldboy/centos:6.8

5）push推送镜像 docker push 10.0.0.11:5000/oldboy/centos:6.8