Docker培训

学习参考资料

注意：学习技术，最好的是官方文档，而且要多练多操作

Docker官网：
http://www.docker.com
https://hub.docker.com
https://docs.docker.com/install/linux/docker-ce/centos/

参考资料：
1、《Docker从入门到实践》
2、菜鸟教程：
http://www.runoob.com/docker/docker-tutorial.html

一、Docker简介

1、简介

容器定义
容器是一种轻量级、可移植、自包含的软件打包技术，使应用程序可以在几乎任何地方以相同的方式运行。开发人员在自己笔记本上创建并测试好的容器，无需任何修改就能够在生产系统的虚拟机、物理服务器或公有云主机上运行。

Docker 是一个开源的应用容器引擎，基于 Go 语言 并遵从Apache2.0协议开源。

Docker 可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。

容器是完全使用沙箱机制，相互之间不会有任何接口（类似 iPhone 的 app）,更重要的是容器性能开销极低。

Docker 从 17.03 版本之后分为 CE（Community Edition: 社区版） 和 EE（Enterprise Edition: 企业版），我们用社区版就可以了。

2、Docker的应用场景

1）简化配置
虚拟机的最大好处是能在你的硬件设施上运行各种配置不一样的平台(软件, 系统), Docker在降低额外开销的情况下提供了同样的功能. 它能让你将运行环境和配置放在代码汇总然后部署, 同一个Docker的配置可以在不同的环境环境中使用, 这样就降低了硬件要求和应用环境之间耦合度.

2）代码流水线管理
代码从开发者的机器到最终在生产环境上的部署, 需要经过很多的中坚环境. 而每一个中间环境都有自己微小的差别, Docker给应用提供了一个从开发到上线均一致的环境, 让代码的流水线变得简单不少.

3）提升开发效率
不同环境中, 开发者的共同目标:
a.想让开发环境尽量贴近生产环境.
b.想快速搭建开发环境

开发环境的机器通常内存比较小, 之前使用虚拟的时候, 我们经常需要为开发环境的机器加内存, 而现在Docker可以轻易的让几十个服务在Docker中跑起来.

4）隔离应用
开发时会在一个台机器上运行不同的应用.
a.为了降低成本, 进行服务器整合
b.将一个整体式的应用拆分成低耦合的单个服务(微服务架构)

5）整合服务器
Docker隔离应用的能力使得Docker可以整合多个服务器以降低成本. 由于没有多个操作系统的内存占用, 以及能在多个实例之间共享没有使用的内存, Docker可以比虚拟机提供更好的服务器整合解决方案.

6）调试能力
Docker提供了很多的工具, 这些工具不一定只是针对容器, 但是却适用于容器. 他们提供了很多功能, 包括可以为容器设置检查点, 设置版本, 查看两个容器之间的差别, 这些特性可以帮助调试Bug.

7）多租户环境
多租户环境的应用中, 它可以避免关键应用的重写.我们一个特别的关于这个场景的例子是为IoT(物联网)的应用开发一个快速, 易用的多租户环境. 这种多租户的基本代码非常复杂, 很难处理, 重新规划以应用不但消耗时间, 也浪费金钱.
使用Docker, 可以为每一个租户的应用层的多个实例创建隔离的环境, 这不仅简单而且成本低廉, 因为Docker环境启动的速度快, diff命令很高效.

8）快速部署
Docker为进程创建一个容器, 不需要启动一个操作系统, 时间缩短为秒级别.
可以在数据中心创建销毁资源而无须担心重新启动带来的开销. 通常数据中心的资源利用率只有30% , 通过使用Docker并进行有效的资源分配可以提高资源的利用率.

3、Docker 的优点

1）简化程序：
Docker让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，便可以实现虚拟化。Docker改变了虚拟化的方式，使开发者可以直接将自己的成果放入Docker中进行管理。方便快捷已经是 Docker的最大优势，过去需要用数天乃至数周的　　任务，在Docker容器的处理下，只需要数秒就能完成。

2）避免选择恐惧症：
如果你有选择恐惧症，还是资深患者。Docker帮你打包你的纠结！比如Docker镜像；Docker镜像中包含了运行环境和配置，所以Docker可以简化部署多种应用实例工作。比如Web应用、后台应用、数据库应用、大数据应用比如Hadoop集群、消息队列等等都可以打包成一个镜像部署。

3）节省开支：
一方面，云计算时代到来，使开发者不必为了追求效果而配置高额的硬件，Docker改变了高性能必然高价格的思维定势。Docker与云的结合，让云空间得到更充分的利用。不仅解决了硬件管理的问题，也改变了虚拟化的方式。

3、docker的缺点

Docker的安装非常容易.目前,Docker支持所有的Linux系列系统,(Ubuntu,RHEL,Debian等).通过Boot2Docker虚拟工具,在OS X和Windows下也能够正常运行Docker.
注: Docker运行环境的限制:
1）必须是64位机器上运行,目前仅支持 x86_64和AMD64,32系统不支持.
2)系统的Linux内核必须是3.8或者更高,内核支持Device Mapper,AUFS,VFS,btrfs等存储格式.
3)内核必须支持cgroups和命名空间.

4、docker的相关知识点

Docker组件（核心）：镜像、容器、库。

采用C/S架构：客户端（执行程序）→通过命令行和API形式和守候程序（提供Docker服务）进行通讯。

镜像：一个只读的静态模板（框架体系）；包含环境和应用执行代码（框架语言）；采用分层机制。将新增数据通过联合文件系统附加在原基础上。

容器：一个运行时环境，是镜像的运行状态，是镜像执行的一种动态表现。

库：使用注册服务器（共有的or私有的）存储和共享用户的镜像，是某个特定用户存储镜像的目录。

*Docker特性：

*隔离性：libcontainer（默认容器）=>内核命名空间。

*安全性：内部（cgroups）| 容器和宿主主机（内核能力机制）。

*可度量性：cgroups（控制组）→资源度量和分配（用户）。

*移植性：AUFS（快速更新）→层的概念。

-----使用AUFS作为Docker容器的文件系统，提供的好处：
1）节省存储空间:多个容器可以共享一个基础镜像存储.
2）快速部署:当要部署多个来自同一个基础镜像的容器时,避免多次复制操作.
3）升级方便:升级一个基础镜像即可影像到所有基于它的容器.
4）增量修改:可以在不改变基础镜像的同时修改其他目录文件,所有的操作都发生在最上层的写操作层,增加了基础镜像的可共享内容.

二、准备

1、使用Centos7系统

注意：root用户执行命令不用加sudo

1)设置普通用户的sudo权限
yum install -y sudo
chmod u+w /etc/sudoers
echo 'wym ALL=(ALL) ALL' >> /etc/sudoers

2)检查selinux和防火墙是否关闭

改配置文件禁用SELINUX
# vim /etc/selinux/config
SELINUX=disabled

setenforce 0 #临时关闭
getenforce #查看状态

systemctl disable firewalld.service #禁止防火墙自启动
systemctl stop firewalld.service #停止防火墙
systemctl status firewalld.service #查看防火墙状态

2、更换阿里yum源，需要连外网

建立备份目录
sudo mkdir -p /etc/yum.repos.d/repobak
备份原有文件
sudo mv /etc/yum.repos.d/*.repo /etc/yum.repos.d/repobak
拉取阿里源文件
sudo curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo
sudo curl -o /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-7.repo
sudo curl -o /etc/yum.repos.d/docker-ce.repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

清除并生成缓存
sudo yum clean all && yum makecache
查看
#sudo yum repolist

3、公司内网yum源

自己搭建的内网yum源配置文件
sudo curl http://192.168.102.3/CentOS-YUM/centos/repo/CentOS-7.repo > /etc/yum.repos.d/Centos-7.repo
sudo curl http://192.168.102.3/CentOS-YUM/centos/repo/epel-7.repo > /etc/yum.repos.d/epel-7.repo
sudo curl http://192.168.102.3/CentOS-YUM/centos/repo/docker-ce1806.repo >

```
/etc/yum.repos.d/docker-ce.repo
# sudo yum clean all && yum makecache
```

三、安装Docker

1、官方安装

```
官方文档:
https://docs.docker.com/install/linux/docker-ce/centos/

sudo yum install -y yum-utils device-mapper-persistent-data lvm2
sudo yum repolist
sudo yum install epel-release  #安装第三方源
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

sudo yum list docker-ce --showduplicates | sort -r
sudo yum install docker-ce docker-ce-cli containerd.io

# yum install -y docker-ce-18.06.1.ce
# yum install -y docker-ce-18.09.3-3.el7

sudo systemctl start docker
sudo systemctl enable docker

查看是否安装Docker软件包
# yum list installed | grep docker
# rpm -qa|grep docker
# docker version


$ curl -fsSL https://get.docker.com -o get-docker.sh
$ sudo sh get-docker.sh

如果您想将Docker用作非root用户，您现在应该考虑将您的用户添加到“docker”组，例如:
sudo groupadd docker
sudo usermod -aG docker $USER

$ sudo yum remove docker-ce
$ sudo rm -rf /var/lib/docker
```

1.1安装低版本，解决依赖问题

```
通过这个地址我们查看和我们安装docker版本一致的rpm包
https://download.docker.com/linux/centos/7/x86_64/stable/Packages/

要先安装docker-ce-selinux-17.03.2.ce，否则安装docker-ce会报错
sudo yum install -y
https://download.docker.com/linux/centos/7/x86_64/stable/Packages/docker-ce-selinux-17.03.2.ce-1.el7.centos.noarch.rpm
```

然后再安装 docker-ce-17.03.2.ce，就能正常安装
sudo yum -y install docker-ce-17.03.2.ce docker-ce-cli-17.03.2.ce containerd.io

2、离线安装包

请注意：--downloadonly参数将自动下载程序包安装时所需要的所有依赖，所以建议在全新的系统中使用本命令，因为在已经安装过部分依赖的系统上，yum不会将所有需要的依赖下载完全，例如下面这幅图所示

1）使用--downloadonly参数之后，yum在下载完程序包后就会显示一句“exiting because "Download Only" specified”并自动退出，此时要下载的程序包已经被放置到了yum的默认存放位置，在CentOS 7 x64下，这个默认路径是：

/var/cache/yum/x86_64/7/<repo>/packages/

2）如果要指定yum的下载目录，还需要一个“--downloaddir”参数，例如我要将nginx程序包下载到当前文件夹下，就使用下面的命令：

yum install -y --downloadonly --downloaddir=. nginx

下载离线的docker安装包
yum install -y --downloadonly --downloaddir=/rpmpackage/docker-ce docker-ce-18.06.1.ce

3）yum离线安装
在刚刚第二步中我们使用了yum --downloadonly命令离线下载了想要的程序包安装文件，现在我们有一台因为种种原因而不能上网的系统，我们现在要把刚刚下载下来的程序包安装到这台电脑上该怎么办呢?

很简单，用yum localinstall命令。

首先将我们下载下来的程序包及其依赖去使用U盘等方式拷贝到这台不能上网的电脑中，然后进入程序包存放目录，执行下面的命令（以supervisor为例）：

```
yum localinstall -y --nogpgcheck supervisor-3.1.4-1.el7.noarch.rpm \
    python-backports-1.0-8.el7.x86_64.rpm \
    python-backports-ssl_match_hostname-3.4.0.2-4.el7.noarch.rpm \
    python-meld3-0.6.10-1.el7.x86_64.rpm \
    python-setuptools-0.9.8-7.el7.noarch.rpm
```

1.使用yum localinstall命令需要的程序包时需要同时安装程序包所有的依赖项目，否则还是会尝试联网去下载缺少的依赖项目；
2.“--nogpgcheck”参数主要是为了不让yum对程序包进行GPG验证；
3.除了yum localinstall命令以外，还可以使用rpm -ivh命令安装rpm包，这里不再单独讨论。

或者构建本地yum库来安装。

需要说明的是，为了在离线环境建软件源，createrepo是必不可少的模块，因此需要下载createrepo相关模块。

yum install --downloadonly --downloaddir=/rpmpackage/createrepo createrepo

一般会下载三个包，一个是createrepo，另外两个是依赖包。

```
# rpm -ivh python-deltarpm-3.6-3.el7.x86_64.rpm
# rpm -ivh createrepo-0.9.9-28.el7.noarch.rpm
```

centos7.4默认已经安装的
deltarpm-3.6-3.el7.x86_64
libxml2-python-2.9.1-6.el7_2.3.x86_64

createrepo构建本地软件源

假设安装包在目标机的/rpmpackage/createrepo目录下。

```
# createrepo  /rpmpackage/docker-ce/
```

修改yum软件源

移除现有的软件源
```
# mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo_bk
```

开启本地软件源
```
# vim /etc/yum.repos.d/CentOS-Media.repo
[docker-ce]
name=local docker-ce
baseurl=file:////rpmpackage/docker-ce/
enabled=1
gpgcheck=0
```

这样就可以使yum采用本地源安装软件。

输入yum repolist看是否可以看到自己构建的本地源

清除缓存 #yum clean all
创建缓存 #yum makecache

查看本地源是否成功，通过yum list是否输出新的rpm包。查询到则证明成功
查看包信息
```
# yum info docker-ce
```

在目标机安装目标软件
使用yum正常安装软件即可。

```
# yum install -y docker-ce
# yum install -y docker-ce-18.06.1.ce #docker-ce-18.09.2
```

如果用的是纯净的虚拟机环境，并且和目标机保持一致，那么依赖包就会都安装，yum安装就会很顺利。除非个别包会有依赖冲突，A依赖B，B又依赖A，导致无法安装，此时可以用rpm命令强制安装其中一个，再用yum安装软件即可。

```
# rpm -ivh demo.rpm --nodeps --force
```

如果安装中出现类似下面的错误：

```
 Package fglrx-glc22-4.1.0-3.2.5.i586.rpm is not signed

需要加个 --nogpgcheck 参数。
# yum localinstall fglrx-glc22-4.1.0-3.2.5.i586.rpm --nogpgcheck
```

3、选择需要安装docker版本

```
# sudo yum install -y yum-utils device-mapper-persistent-data lvm2
# sudo yum list docker-ce --showduplicates
# sudo yum install -y docker-ce 或安装特定版本

# yum install -y docker-ce-18.09.3-3.el7

# sudo systemctl start docker
# sudo systemctl enable docker

# yum list docker-ce --showduplicates
Failed to set locale, defaulting to C
Loaded plugins: fastestmirror
Determining fastest mirrors
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
Installed Packages
docker-ce.x86_64            18.03.1.ce-1.el7.centos           @docker-ce-stable
Available Packages
docker-ce.x86_64            17.03.0.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            17.03.1.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            17.03.2.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            17.03.3.ce-1.el7                  docker-ce-stable
docker-ce.x86_64            17.06.0.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            17.06.1.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            17.06.2.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            17.09.0.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            17.09.1.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            17.12.0.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            17.12.1.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            18.03.0.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            18.03.1.ce-1.el7.centos           docker-ce-stable
docker-ce.x86_64            18.06.0.ce-3.el7                  docker-ce-stable
docker-ce.x86_64            18.06.1.ce-3.el7                  docker-ce-stable
docker-ce.x86_64            18.06.2.ce-3.el7                  docker-ce-stable
docker-ce.x86_64            18.06.3.ce-3.el7                  docker-ce-stable
docker-ce.x86_64            3:18.09.0-3.el7                   docker-ce-stable
docker-ce.x86_64            3:18.09.1-3.el7                   docker-ce-stable
docker-ce.x86_64            3:18.09.2-3.el7                   docker-ce-stable
docker-ce.x86_64            3:18.09.3-3.el7                   docker-ce-stable
docker-ce.x86_64            3:18.09.4-3.el7                   docker-ce-stable
docker-ce.x86_64            3:18.09.5-3.el7                   docker-ce-stable
```

4、删除Docker

```
# yum remove docker-ce
# rm -rf /var/lib/docker

# yum -y remove docker docker-common docker-selinux docker-engine docker-engine-selinux
container-selinux docker-ce

# yum -y remove docker-ce-cli


查看是否安装Docker软件包
# yum list installed | grep docker
# rpm -qa|grep docker
```

5、镜像加速

```
鉴于国内网络问题，后续拉取Docker镜像十分缓慢，我们可以需要配置加速器来解决。

新版的Docker使用/etc/docker/daemon.json(Linux）来配置守护进程。

请在该配置文件中加入（没有该文件的话，请先建一个）：

如果没有，就建立docker文件夹
sudo mkdir -p /etc/docker

可以选择多个加速器，建议阿里，但是需要注册
# https://xxxx.mirror.aliyuncs.com
# http://hub-mirror.c.163.com
# http://f1361db2.m.daocloud.io

默认没有daemon.json,自己用下面的一种方法新建
# vi /etc/docker/daemon.json
{
  "registry-mirrors": ["https://xxxx.mirror.aliyuncs.com"]
}

"registry-mirrors": ["https://docker.mirrors.ustc.edu.cn","https://al9ikvwc.mirror.aliyuncs.com"]

或
# sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["http://hub-mirror.c.163.com"]
}
EOF

或
# sudo echo -e "{\n \"registry-mirrors\": [\"http://hub-mirror.c.163.com\"]\n}" >
/etc/docker/daemon.json
```

```
#sudo echo -e '{"registry-mirrors": ["https://7bezldxe.mirror.aliyuncs.com/"],"insecure-registries": ["http://192.168.113.38"]}' > /etc/docker/daemon.json
或
# sudo curl -sSL https://get.daocloud.io/daotools/set_mirror.sh | sh -s http://f1361db2.m.daocloud.io

注意查看格式是否正确
# cat /etc/docker/daemon.json

重新加载并启动
# systemctl daemon-reload
# systemctl restart docker
```

6、Windows Docker 安装

```
win7、win8 等需要利用 docker toolbox 来安装，国内可以使用阿里云的镜像来下载，下载地址：http://mirrors.aliyun.com/docker-toolbox/windows/docker-toolbox/

docker toolbox 是一个工具集，它主要包含以下一些内容：
  Docker CLI 客户端，用来运行docker引擎创建镜像和容器
  Docker Machine. 可以让你在windows的命令行中运行docker引擎命令
  Docker Compose. 用来运行docker-compose命令
  Kitematic. 这是Docker的GUI版本
  Docker QuickStart shell. 这是一个已经配置好Docker的命令行环境
  Oracle VM Virtualbox. 虚拟机
```

四、docker常用命令

1、docker命令

```
docker  # docker 命令帮助

Commands:
  attach    Attach to a running container            # 当前 shell 下 attach 连接指定运行镜像
  build     Build an image from a Dockerfile         # 通过 Dockerfile 定制镜像
  commit    Create a new image from a container's changes # 提交当前容器为新的镜像
  cp        Copy files/folders from the containers filesystem to the host path
        # 从容器中拷贝指定文件或者目录到宿主机中
  create    Create a new container                   # 创建一个新的容器，同 run，但不启动容器
  diff      Inspect changes on a container's filesystem  # 查看 docker 容器变化
  events    Get real time events from the server     # 从 docker 服务获取容器实时事件
  exec      Run a command in an existing container   # 在已存在的容器上运行命令
  export    Stream the contents of a container as a tar archive
        # 导出容器的内容流作为一个 tar 归档文件[对应 import ]
  history   Show the history of an image             # 展示一个镜像形成历史
  images    List images                              # 列出系统当前镜像
  import    Create a new filesystem image from the contents of a tarball
        # 从tar包中的内容创建一个新的文件系统映像[对应 export]
```

```
  info     Display system-wide information            # 显示系统相关信息
  inspect  Return low-level information on a container  # 查看容器详细信息
  kill     Kill a running container                   # kill 指定 docker 容器
  load     Load an image from a tar archive           # 从一个 tar 包中加载一个镜像[对应 save]
  login    Register or Login to the docker registry server
           # 注册或者登陆一个 docker 源服务器
  logout   Log out from a Docker registry server      # 从当前 Docker registry 退出
  logs     Fetch the logs of a container              # 输出当前容器日志信息
  port     Lookup the public-facing port which is NAT-ed to PRIVATE_PORT
           # 查看映射端口对应的容器内部源端口
  pause    Pause all processes within a container     # 暂停容器
  ps       List containers                            # 列出容器列表
  pull     Pull an image or a repository from the docker registry server
           # 从docker镜像源服务器拉取指定镜像或者库镜像
  push     Push an image or a repository to the docker registry server
           # 推送指定镜像或者库镜像至docker源服务器
  restart  Restart a running container                # 重启运行的容器
  rm       Remove one or more containers              # 移除一个或者多个容器
  rmi      Remove one or more images
           # 移除一个或多个镜像[无容器使用该镜像才可删除，否则需删除相关容器才可继续或 -f 强制删除]
  run      Run a command in a new container
           # 创建一个新的容器并运行一个命令
  save     Save an image to a tar archive             # 保存一个镜像为一个 tar 包[对应 load]
  search   Search for an image on the Docker Hub      # 在 docker hub 中搜索镜像
  start    Start a stopped containers                 # 启动容器
  stop     Stop a running containers                  # 停止容器
  tag      Tag an image into a repository             # 给源中镜像打标签
  top      Lookup the running processes of a container  # 查看容器中运行的进程信息
  unpause  Unpause a paused container                 # 取消暂停容器
  version  Show the docker version information        # 查看 docker 版本号
  wait     Block until a container stops, then print its exit code
           # 截取容器停止时的退出状态值
Run 'docker COMMAND --help' for more information on a command.

docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
OPTIONS说明：
-a stdin: 指定标准输入输出内容类型，可选 STDIN/STDOUT/STDERR 三项；
-d: 后台运行容器，并返回容器ID；
-i: 以交互模式运行容器，通常与 -t 同时使用；
-p: 端口映射，格式为：主机(宿主)端口:容器端口
-t: 为容器重新分配一个伪输入终端，通常与 -i 同时使用；
--name="nginx-lb": 为容器指定一个名称；
--dns 8.8.8.8: 指定容器使用的DNS服务器，默认和宿主一致；
--dns-search example.com: 指定容器DNS搜索域名，默认和宿主一致；
-h "mars": 指定容器的hostname；
-e username="ritchie": 设置环境变量；
--env-file=[]: 从指定文件读入环境变量；
--cpuset="0-2" or --cpuset="0,1,2": 绑定容器到指定CPU运行；
-m :设置容器使用内存最大值；
--net="bridge": 指定容器的网络连接类型，支持 bridge/host/none/container: 四种类型；
--link=[]: 添加链接到另一个容器；
--expose=[]: 开放一个端口或一组端口；
```

```
实例：
# docker run -itd --name box1 busybox
# docker ps
# docker exec -it 802 /bin/sh
```

2、docker常用命令

```
$ sudo docker #显示docker命令
$ sudo docker -v #显示 Docker 版本信息
$ sudo docker version #显示 Docker 版本信息
$ sudo docker info #显示 Docker 系统信息，包括镜像和容器数

$ sudo docker search centos
$ sudo docker search centos --no-trunc #可显示完整的镜像描述
$ sudo docker search -s 10 tomcat   # 搜索10星以上的tomcat镜像

$ sudo docker pull tomcat #拉取指定镜像，可以加版本
$ sudo docker pull nginx:1.15.9-alpine
$ sudo docker run -itd --name nginx1 -p 8081:80 nginx:1.15.9-alpine

$ sudo docker images #显示所有镜像

把镜像保存成一个tar文件，注意如果目录没有，需要提前建立一下，docker不会帮你建立目录的
$ sudo docker save 镜像名 -o /imagesbak/xxx.tar

筛选关键字批量打包
# docker save $(docker images | grep 关键字 | awk 'BEGIN{OFS=":";ORS=" "}{print $1,$2}') -o
/imagesbak/dockerimages.tar

加载镜像备份
$ sudo docker load -i /imagesbak/xxx.tar

$ sudo docker rmi 镜像ID

拷贝docker容器里的文件
$ docker cp nginx1:/etc/nginx/nginx.conf /home/w/tool/nginx/conf/nginx.conf

$ sudo docker ps #列出所有运行中容器
$ sudo docker ps -a #列出所有容器（含沉睡镜像）
$ sudo docker ps -l #仅列出最新创建的一个容器
$ sudo docker ps -n=3 #列出最近创建的3个容器
$ sudo docker ps -s #显示容器大小

$ sudo docker stop 停止容器名或ID
$ sudo docker start 启动容器名或ID
$ sudo docker restart 重启容器名或ID

进入容器
$ sudo docker exec -it 容器ID /bin/sh
```

```
# 进入docker 容器的小脚本

下面是一个通过nsenter进入docker容器的例子脚本：
文件名字：ns
使用方法：将文件放入系统PATH路径下，进入容器方式ns <container-name/container-id>

如果docker容器没有提供ssh,那么进入docker容器的方法，一般是 attach ,exec,nsenter
attach 进入后再退出，会引起docker 容器停止。exec 每次输入比较麻烦。
比较方便的是用 nsenter . nsenter 进入需要查docker 容器的pid 。所以，写了下面的脚本，方便进入。

#!/bin/bash
docker ps
echo "====================================\r"
read -p "input docker name:" did
PID=$(docker inspect --format "{{.State.Pid}}" $did)
nsenter --target $PID --mount --uts --ipc --net --pid

该脚本会提示当前运行的docker容器，然后输入docker 的id 后，就进入了docker容器


删除，容器要先停止
$ sudo docker rm 容器ID
-f 强行移除该容器，即使其正在运行；
-l 移除容器间的网络连接，而非容器本身；
-v 移除与容器关联的空间。

查看
$ sudo docker top #查看一个正在运行容器进程
$ sudo docker inspect 镜像或容器ID  #检查镜像或者容器的参数，默认返回 JSON 格式
$ sudo docker logs -f 容器ID #查看指定容器的日志记录

$ docker stats 容器ID #查看容器的性能
$ docker system df #查看Docker的磁盘使用情况
$ sudo du -hs /var/lib/docker/ #查看docker目录空间


其他命令

#获取Container IP地址（Container状态必须是Up）
$ sudo docker inspect 容器ID | grep IPAddress | cut -d '"' -f 4

#获取环境变量
$ sudo docker exec 677 env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=3863f1da45f4
HOME=/root
```

3、批量操作docker命令

```
筛选关键字
# docker save $(docker images | grep geerlingguy | awk 'BEGIN{OFS=":";ORS=" "}{print $1,$2}') -o
/home/wym/wymdockerimagesgeerlingguy.tar

批量删除没有启动的容器
# docker rm $(docker ps --all -q -f status=exited)

删除没有使用的镜像
# docker rmi -f $(docker images | grep "<none>" | awk "{print \$3}")

docker批量删除容器、镜像
1、删除所有容器
# docker rm `docker ps -a -q`
2、删除所有镜像
# docker rmi --force `docker images -q`
3、按条件删除镜像
    没有打标签
# docker rmi `docker images -q | awk '/^<none>/ { print $3 }'`

    镜像名包含关键字
# docker rmi --force `docker images | grep doss-api | awk '{print $3}'`   //其中doss-api为关键字

批量建立100个docker容器，并绑定端口
# for i in 'seq 2201 2299';do docker run -itd -p $i:22 -p 1$i:80 centos68 /bin/bash

批量开启容器的sshd服务
# for i in 'docker ps -aq';do docker exec $i /etc/init.d/sshd start;done

筛选出ubuntu容器，print $1表示输出第一列
# docker ps -a|grep ubuntu|awk '{print $1}'
筛选出ubuntu容器，并停止
# docker ps -a|grep ubuntu|awk '{print $1}'|xargs docker stop
筛选出ubuntu容器，并删除
# docker ps -a|grep ubuntu|awk '{print $1}'|xargs docker rm
筛选出ubuntu镜像ID，print $1表示输出第三列
# docker images|grep ubuntu|awk '{print $3}'
筛选出ubuntu镜像ID，并删除镜像
# docker images|grep ubuntu|awk '{print $3}'|xargs docker rmi
```

六、Docker实战

1、docker使用tomcat环境

```
# docker pull tomcat:8.5.38-jre8-alpine
# docker inspect tomcat:8.5.38-jre8-alpine  #查看镜像信息，比如查看tomcat的路径
CATALINA_HOME=/usr/local/tomcat

启动容器，映射到主机8018端口，挂载/home/wym/tomcat/test目录，如果没有，则会自动创建
# docker run -dit -p 8018:8080 --name tomcat01 -v
/home/wym/tomcat/test:/usr/local/tomcat/webapps/test tomcat:8.5.38-jre8-alpine
```

```
$ docker ps -l #查看最新的容器
$ docker exec -it 677 /bin/bash  #进入容器

查看tomcat
http://192.168.113.37:8018/

查看相关页面
http://192.168.113.37/test/helloworld.html

在本机操作，编写helloworld.html文件，容器里同时也有，页面也同时显示
# cd /home/wym/tomcat/test
# echo "<h1> Hello World !<h1>" > hello.html
```

2、docker使用mysql

```
# docker pull mysql:5.7
# docker run --name mysql01 -p 3306:3306 -e MYSQL_ROOT_PASSWORD=newcapec -d
mysql:5.7.25

# docker inspect mysql01
# docker exec mysql01 env

使用数据库工具就可以远程连接mysql01，主机IP，端口3306，用户名root，密码newcapec

# docker exec -it mysql01 /bin/sh
# mysql -u root -p
mysql> show databases;
```

七、使用docker-compose

1、安装

```
https://docs.docker.com/compose/install/

在Linux系统上安装Compose
在Linux上，您可以从GitHub上的Compose存储库发行页面下载Docker Compose二进制文件。按照
链接中的说明进行操作，该链接涉及curl在终端中运行命令以下载二进制文件。这些分步说明也包含在
下面。

运行此命令以下载Docker Compose的当前稳定版本：

# sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
要安装不同版本的Compose，请替换1.24.0 为要使用的Compose版本。

对二进制文件应用可执行权限：
# sudo chmod +x /usr/local/bin/docker-compose
注意：如果docker-compose安装后命令失败，请检查您的路径。您还可以创建/usr/bin路径中的符号
```

链接或任何其他目录。

例如：
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

测试安装。
docker-compose --version
docker-compose version 1.24.0, build 1110ad01

2、常用docker-compose命令

首先要进入docker-compose.yml所在目录，再执行命令

其它命令（修改配置文件后最好把容器删除再创建）：
```
docker-compose up -d          ###后台启动，如果容器不存在根据镜像自动创建
docker-compose down -v        ###停止容器并删除容器
docker-compose start          ###启动容器，容器不存在就无法启动，不会自动创建镜像
docker-compose stop           ###停止容器
docker-compose logs       ###查看日志（harbor日志存放地址 /var/logs/harbor）
docke-compose ps          ###查看容器
```

如果用的不是标准yml名称docker-compose.yml，就要用-f参数指定自定义的yml文件
$ docker-compose -f docker-wordpress.yml ps

3、docker-compose 命令

```
Commands:
  build              Build or rebuild services #构建或重建服务
  bundle             Generate a Docker bundle from the Compose file
  config             Validate and view the compose file
  create             Create services
  down               Stop and remove containers, networks, images, and volumes
  events             Receive real time events from containers
  exec               Execute a command in a running container
  help               Get help on a command #命令帮助
  kill               Kill containers #杀掉容器
  logs               View output from containers #显示容器的输出内容
  pause              Pause services
  port               Print the public port for a port binding #打印绑定的开放端口
  ps                 List containers #显示容器
  pull               Pull service images #拉取服务镜像
  push               Push service images
  restart            Restart services #重启服务
  rm                 Remove stopped containers #删除停止的容器
  run                Run a one-off command #运行一个一次性命令
  scale              Set number of containers for a service #设置服务的容器数目
  start              Start services #开启服务
  stop               Stop services #停止服务
```

```
top            Display the running processes
unpause        Unpause services
up             Create and start containers #创建并启动容器
version        Show the Docker-Compose version information
```

3、例子

```
docker-compose 如何配置
先看看我自己写的一个 docker-compose.yml

version: '2'
services:
  nginx:
      image: bitnami/nginx:latest
      ports:
        - '80:80'
        - '1443:443'
      volumes:
        - /root/wp_yunlan/nginx/:/bitnami/nginx/
  mariadb:
      image: bitnami/mariadb:latest
      volumes:
        - /root/wp_yunlan/mariadb:/bitnami/mariadb
  wordpress:
      image: bitnami/wordpress:latest
      depends_on:
        - mariadb
        - nginx
      environment:
        - WORDPRESS_USERNAME=neptunemoon   #这个账户你是自己设定的
        - WORDPRESS_PASSWORD=123123       #这个密码是你自己设定的
      ports:
        - '8080:80'
        - '8081:443'
      volumes:
        - /root/wp_yunlan/wordpress:/bitnami/wordpress
        - /root/wp_yunlan/apache:/bitnami/apache
        - /root/wp_yunlan/php:/bitnami/php
nginx 和 mariadb，wordpress 是要启动的三个服务

顺序不是重要的,我们看见wordpress中有个 depends_on: 的属性

depends_on: 依赖
代表wordpress 依赖于
- mariadb
- nginx
两个服务， 所以他们两个会先启动

image: 镜像，就是你的 docker 镜像

# docker search mariadb 找到我们需要的镜像
```

八、Docker私有仓库使用说明

https://github.com/goharbor/harbor/releases

https://storage.googleapis.com/harbor-releases/release-1.7.0/harbor-offline-installer-v1.7.5.tgz

https://github.com/goharbor/harbor/blob/release-1.8.0/docs/installation_guide.md

1.安装harbor

安装步骤归结为以下内容

1）下载安装程序：
可以从发布页面下载安装程序的二进制文件。选择在线或离线安装程序。使用tar命令提取包。

在线安装：
```
$ tar xvf harbor-online-installer-<version>.tgz
```
脱机安装：
```
$ tar xvf harbor-offline-installer-<version>.tgz
```

2）配置harbor.cfg；
必需参数：
hostname：目标主机的主机名或IP地址，比如192.168.113.37
harbor_admin_password：管理员的初始密码。此密码仅在Harbor首次启动时生效。之后，将忽略此设置，并且应在Portal中设置管理员密码。请注意，默认用户名/密码为admin/Harbor12345。
```
$ sudo vi harbor.cfg
```

3）运行install.sh安装并启动Harbor;
一旦harbor.cfg和存储后端（可选）配置，安装和使用的启动港install.sh脚本。请注意，在线安装程序可能需要一些时间才能从Docker hub下载Harbor映像。

```
$ sudo ./install.sh
```

4）登录http://192.168.113.37，使用admin登录

5）要更改Harbour的配置，请先停止现有的Harbor实例并进行更新harbor.cfg。然后运行prepare脚本以填充配置。最后重新创建并启动Harbor的实例：

```
$ sudo docker-compose down -v
$ vim harbor.cfg
$ sudo prepare
$ sudo docker-compose up -d
```

2.使用harbor

1）访问路径：http://192.168.102.37
2）首先，自己注册，然后登录，密码必须包含大小写、数字
3）登录后，可以看到公开的项目和自己的私有项目，可以自己新建私有项目
4）本地的Docker配置镜像仓库地址

问题：
docker1.3.2版本开始默认docker registry使用的是https，我们设置Harbor默认http方式，所以当执行用docker login、pull、push等命令操作非https的docker regsitry的时就会报错。

解决办法：

- 1)、 如果系统是MacOS，则可以点击“Preference”里面的“Advanced”在“InsecureRegistry”里加上192.168.102.37，重启Docker客户端就可以了。
- 2)、 如果系统是Ubuntu，则修改配置文件/lib/systemd/system/docker.service，修改[Service]下ExecStart参数，增加– insecure-registry 192.168.102.37
- 3)、 如果系统是Centos，可以修改配置/etc/sysconfig/docker，将OPTIONS增加 –insecure-registry 192.168.102.37
- 4)、 如果是新版本的docker,在/etc/sysconfig/ 没有docker这个配置文件的情况下,在/etc/docker/目录下，如果没有daemon.json则新建，有了就添加insecure-registries。

```
# sudo vim /etc/docker/daemon.json
{
 "insecure-registries": ["http://192.168.102.37"]
}
```

配置完daemon.json，需要重新加载然后重启

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

登录公开的仓库

公开的仓库不需要登录，在页面查看需要的镜像名称，通过pull命令下载

```
# sudo docker pull 192.168.102.37/library/busybox
```

私有仓库，首先要用自己的账号登录

```
# sudo docker login 192.168.102.37
Username: wym
Password: newcapec
Login Succeeded

# sudo docker login -u admin -p admin 192.168.102.37
```

给要上传的镜像打标签，注意必须包括私有仓库IP、端口和项目名称

```
# sudo docker tag busybox 192.168.102.37/wymproject/busybox
```

上传镜像

```
# sudo docker push 192.168.102.37/wymproject/busybox
```

下载镜像

```
# sudo docker pull 192.168.102.37/wymproject/busybox
```

项目管理员可以添加项目成员，删除镜像等操作

九、管理工具

```
# docker run -d -p 9000:9000 --restart=always -v /var/run/docker.sock:/var/run/docker.sock --name prtainer-test portainer/portainer

http://192.168.113.37:9000 ，登录admin，设置密码admin

镜像weaveworks/scope
scope launch 将以容器方式启动 Weave Scope。
http://127.0.0.1:4040
```

十、docker常见问题

1、CentOS7中Docker-ce的卸载和安装

2、centos 安装docker时出现依赖关系问题的解决办法

```
我们在安装老版本的docker时可能会出现：
“正在处理依赖关系 docker-ce-selinux >= 17.03.0.ce-1.el7.centos，它被软件包 docker-ce-17.03.0.ce-1.el7.centos.x86_64 需要
软件包 docker-ce-selinux 已经被 docker-ce-cli 取代，但是取代的软件包并未满足需求”

等一大串的问题

这时我们需要通过 yum install 安装一个rpm包

通过这个地址我们查看和我们安装docker版本一直的rpm包
https://download.docker.com/linux/centos/7/x86_64/stable/Packages/


要先安装docker-ce-selinux-17.03.2.ce，否则安装docker-ce会报错

yum install https://download.docker.com/linux/centos/7/x86_64/stable/Packages/docker-ce-selinux-17.03.2.ce-1.el7.centos.noarch.rpm

然后再安装 docker-ce-17.03.2.ce，就能正常安装

sudo yum install docker-ce-17.03.2.ce-1.el7.centos

无法安装软件包 docker-ce-selinux-17.03.2.ce-1.el7.centos.noarch 。被已安装软件包 1:docker-ce-cli-18.09.0-3.el7.x86_64 标记为废除
错误：无须任何处理
[root@k8sm ~]# yum -y remove docker-ce-cli
```

3、让docker 容器开机自动启动

docker run 指令中加入 --restart=always 就行。
sudo docker run --restart=always .....

如果创建时未指定 --restart=always ,可通过update 命令设置
docker update --restart=always 容器ID

如果你想取消掉
docker update --restart=no <CONTAINER ID>

4、迁移 /var/lib/docker 目录

/var/lib/docker/overlay2 占用很大，清理Docker占用的磁盘空间，迁移 /var/lib/docker 目录

迁移 /var/lib/docker 目录

4.1 停止docker服务。
systemctl stop docker

4.2 创建新的docker目录，执行命令df -h,找一个大的磁盘。 我在 /home目录下面建了 /home/docker/lib目录，执行的命令是：
mkdir -p /home/docker/lib

4.3 迁移/var/lib/docker目录下面的文件到 /home/docker/lib：
rsync -avz /var/lib/docker /home/docker/lib/

mv /var/lib/docker /home/y/docker/lib/docker/

4.4 配置 /etc/systemd/system/docker.service.d/devicemapper.conf。
查看 devicemapper.conf 是否存在。如果不存在，就新建。
mkdir -p /etc/systemd/system/docker.service.d/
vim /etc/systemd/system/docker.service.d/deivcemapper.conf

4.5 然后在 devicemapper.conf 写入：（同步的时候把父文件夹一并同步过来，实际上的目录应在 /home/docker/lib/docker ）

[Service]
ExecStart=
ExecStart=/usr/bin/dockerd --graph=/home/y/docker/lib/docker

ExecStart=/usr/bin/dockerd --insecure-registry=私服地址 --graph=/home/docker/lib

注意：如果/etc/systemd/system/docker.service.d/devicemapper.conf，这个路径找不到的话，就新建，新建之后加入内容，没有私服地址的话就可以去掉”--insecure-registry=私服地址”。

4.6 重新加载 docker

systemctl daemon-reload
systemctl restart docker
systemctl enable docker

4.7 为了确认一切顺利，运行
docker info
命令检查Docker 的根目录.它将被更改为 /home/y/docker/lib/docker

Docker Root Dir: /home/docker/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/

4.8 启动成功后，再确认之前的镜像还在：
docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
AAA/AAA	v2	7331b8651bcc	27 hours ago	3.85GB
BBB/BBB	v1	da4a80dd8424	28 hours ago	3.47GB

4.9 确定容器没问题后删除/var/lib/docker/目录中的文件。
rm -rf /var/lib/docker

5、配置centos8解决 docker Failed to get D-Bus connection 报错

原因及解决方式：
在创建docker容器时添加--privileged
这个的原因是因为dbus-daemon没能启动。其实systemctl并不是不可以使用。将你的CMD或者entrypoint设置为/usr/sbin/init即可。会自动将dbus等服务启动起来。
然后就可以使用systemctl了。命令如下

docker run -it --name cobbler --privileged=true jasonlix/docker-cobbler /usr/sbin/init

6、不使用sudo命令执行docker

为什么需要创建docker用户组?
Docker守候进程绑定的是一个unix socket，而不是TCP端口。这个套接字默认的属主是root，其他是用户可以使用sudo命令来访问这个套接字文件。因为这个原因，docker服务进程都是以root帐号的身份运行的。
为了避免每次运行docker命令的时候都需要输入sudo，可以创建一个docker用户组，并把相应的用户添加到这个分组里面。当docker进程启动的时候，会设置该套接字可以被docker这个分组的用户读写。这样只要是在docker这个组里面的用户就可以直接执行docker命令了。
警告：该dockergroup等同于root帐号，具体的详情可以参考这篇文章：Docker Daemon Attack Surface.
操作步骤：
使用有sudo权限的帐号登录系统。
创建docker分组，并将相应的用户添加到这个分组里面。
sudo usermod -aG docker your_username
退出，然后重新登录，以便让权限生效。
确认你可以直接运行docker命令。
$ docker run hello-world

7、Docker 容器中运行 Docker 命令

```
# docker run -dit --name ubuntu1604 ubuntu:16.04

通常您可以通过安装Docker套接字从容器内管理主机容器。
# docker run -it -v /var/run/docker.sock:/var/run/docker.sock --name ubuntu1604 ubuntu:16.04
sh -c "apt-get update ; apt-get install docker.io -y ; bash"

# docker run -dit --name c761810 --privileged=true centos:7.6.1810 /usr/sbin/init

Docker里运行Docker docker in docker(dind)

Docker 容器中运行 Docker 命令
在使用 GitLab/Jenkins 等 CI 软件的时候需要使用 Docker 命令来构建镜像，需要在容器中使用 Docker 命令；通过将宿主机的 Docker 共享给容器即可

在启动容器时添加以下命令：
    --privileged \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v $(which docker)r:/bin/docker \
--privileged 表示该容器真正启用 root 权限
-v /var/run/docker.sock:/var/run/docker.sock和-v $(which docker)r:/bin/docker命令将相关的 Docker 文件挂载到容器

# docker run -dit --name c761810 --privileged=true -v /var/run/docker.sock:/var/run/docker.sock -
v $(which docker)r:/bin/docker centos:7.6.1810 /usr/sbin/init

# docker rm -f 容器ID
```

初始化常用运行命令

```
配置SSH登录，关闭SElinux和防火墙
# sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/g' /etc/ssh/sshd_config &&
systemctl restart sshd && systemctl stop firewalld && systemctl disable firewalld.service &&
setenforce 0 && sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config && sed -i
's/SELINUX=permissive/SELINUX=disabled/g' /etc/selinux/config

配置阿里yum源并安装
# mkdir -p /etc/yum.repos.d/repobak && mv /etc/yum.repos.d/*.repo /etc/yum.repos.d/repobak
&& curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo &&
curl -o /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-7.repo && curl -o
/etc/yum.repos.d/docker-ce.repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-
ce.repo && yum clean all && yum makecache && yum install -y wget vim tree net-tools zip unzip
tmux bash-completion && yum install -y docker-ce

配置内网yum源并安装
# mkdir -p /etc/yum.repos.d/repobak && mv /etc/yum.repos.d/*.repo /etc/yum.repos.d/repobak
&& curl http://192.168.102.3/CentOS-YUM/centos/repo/CentOS-7.repo >
/etc/yum.repos.d/Centos-7.repo && curl http://192.168.102.3/CentOS-YUM/centos/repo/epel-
7.repo > /etc/yum.repos.d/epel-7.repo && curl http://192.168.102.3/CentOS-
YUM/centos/repo/docker-ce1806.repo > /etc/yum.repos.d/docker-ce.repo && yum clean all &&
yum makecache && yum install -y wget vim tree net-tools zip unzip tmux bash-completion dstat
```

```
&& yum install -y docker-ce

启动docker
# systemctl start docker && systemctl enable docker

配置加速器和私有镜像仓库
# mkdir -p /etc/docker && echo -e '{"registry-mirrors":
["https://7bezldxe.mirror.aliyuncs.com"],"insecure-registries": ["http://192.168.102.37"]}' >
/etc/docker/daemon.json && systemctl daemon-reload && systemctl restart docker
```