# Docker容器

1. 简单地说，容器是独立运行的一个或一组应用，以及它们的运行态环境。
2. 对应的，虚拟机可以理解为模拟运行的一整套操作系统（提供了运行态环境和其他系统环境）和跑在上面的应用

# 启动容器

1. 启动容器有两种方式
   1. 基于镜像新建一个容器并启动
   2. 将在停止状态（stopped）的容器重新启动
2. 因为Docker容器是在太轻量级了，很多时候用户都是随时删除和新建容器

**新建并启动**

1. 使用**docker run命令**新建并启动容器
   1. 比如，sudo docker run ubuntu:14.04 /bin/echo ‘Hello World’
   2. 这跟在本地直接运行/bin/echo ‘Hello World’几乎感觉不出任何区别
2. 启动一个bash终端，允许用户进行交互：
   1. 命令为sudo docker run -t -i ubuntu:14.04 /bin/bash
   2. -t选项让Docker分配一个伪终端并绑定到容器的标准输入上
   3. -i选项让容器的标准输入保持打开
   4. 在交互模式下，用户可以通过所创建的终端输入命令，和直接用bash一样
3. 当利用docker run命令创建容器时，Docker在后台运行的标准操作包括：
   1. 检查本地是否存在指定的镜像，不存在就从共有仓库下载
   2. 利用镜像创建并启动一个容器
   3. 分配一个**文件系统**，并在**只读**的镜像层外面**挂载一层可读写层**
   4. 从宿主主机配置的网桥接口中桥接一个虚**拟接口**到容器中
   5. 从地址池配置一个**ip地址**给容器
   6. 执行用户指定的**应用程序**
   7. 执行完毕容器被终止

**启动已终止容器**

1. 可以利用**docker start命令**，直接将一个已经终止的容器启动运行
2. 容器的**核心**为所执行的应用程序，所需要的**资源**都是应用程序运行所必需的；除此之外，并没有其他的资源。
   1. 可以在伪终端利用**ps命令**或**top命令**查看进程信息
   2. 可见，容器中仅运行了指定的bash应用
   3. 这种特点使得Docker对资源的**利用率极高**，是真正的**轻容量级虚拟化**

# 守护态运行

1. 更多时候，需要让Docker容器在后台以守护态形式运行。
2. 可以通过在**docker run命令添加参数-d**实现
3. 可以通过**docker ps命令**查看容器信息
4. 可以通过**docker logs命令**查看容器的输出信息

# 终止容器

1. 可以使用**docker stop命令**终止一个运行中的容器
2. 当Docker容器中指定的应用终结时，容器也**自动终止**。
3. 终止状态的容器可以用**docker ps -a命令**看到
4. 处于终止状态的容器，可以通过**docker start命令**重新启动
5. **docker restart**命令会将一个运行态的容器终止，然后再重新启动它

# 进入容器

**attach命令**

1. **docker attach命令**是Docker自带的命令
   1. 比如sudo docker attach nostalgic\_hypatia
2. 使用attach命令有时并不方便。当多个窗口同时attach到同一容器的时候，所有窗口都会**同步显示**。当某个窗口因命令阻塞时，其他窗口也无法执行操作了

**nsenter命令**

1. 安装：
   1. nsenter工具在until-linux包2.23版本后包含。如果没有，按照以下方法从安装

cd /tmp;

curl <https://www.kernel.org/pub/linux/utils/util-linux/v2.24/util-linux-2.24.tar.gz> | tar -zxf-;

cd util-linux-2.24;

./configure --without-ncurses    
make nsenter && sudo cp nsenter /usr/local/bin

1. 使用：
   1. nsenter可以访问另一个进程的名字空间。
   2. nsenter要正常工作需要有root权限
2. 安装最新版本的util-linux：

wget https://www.kernel.org/pub/linux/utils/util-linux/v2.24/util-linux-2.24.tar.gz;

tar xzvf util-linux-2.24.tar.gz

cd util-linux-2.24

./configure --without-ncurses && make nsenter

sudo cp nsenter /usr/local/bin

1. 为了连接到容器，还需要找到容器的第一个进程的PID，可以通过以下命令获取：

PID=$(docker inspect --format “{{ .State.Pid }}” <container>)

1. 通过这个PID，就可以连接到这个容器：

nsenter --target $PID --mount --uts --ipc --net --pid

1. 更简单的，建议下载.bashrc\_docker，并将以下内容放到.bashrc中

wget -P ~ https://github.com/yeasy/docker\_practice/raw/master/\_local/.bashrc\_docker

echo "[ -f ~/.bashrc\_docker ] && . ~/.bashrc\_docker" >> ~/.bashrc

source ~/.bashrc

1. 这个文件定义了很多方便使用Docker的命令，例如**docker -pid**可以获取某个容器的PID；而**docker -enter**可以进入容器或直接在容器内执行命令

# 导出容器

1. 可以使用**docker export命令**导出本地某个容器，将导出容器快照到本地文件，比如：
   1. 先使用sudo docker ps -a获取容器ID
   2. 再使用sudo docker export containerId > fileName.tar导出

# 导入容器快照

1. 可以使用docker import从容器**快照文件**中再导入为镜像：
   1. 例如cat ubuntu.tar | sudo docker import - test/ubuntu:v1.0
2. 也可以通过指定**URL**或者某个**目录**来导入
   1. 例如sudo docker import <http://example.com/exampleimage.tgz> example/imagerepo
3. **\*注**，用户既可以使用**docker load来导入镜像存储文件**到本地镜像库，也可以使用**docker import来导入一个容器快照**到本地镜像库，二者**区别**在于：
   1. 容器快照文件将丢弃所有历史记录和元数据信息（即仅保存容器当时的快照状态）
   2. 镜像存储文件将保存完整记录，体积也要更大
   3. 从容器快照文件导入时可以重新指定标签等元数据信息

# 删除容器

1. 可以使用**docker rm命令**删除一个处于**终止状态**的容器
   1. 例如sudo docker rm example
2. 如果要删除一个运行中的容器，可以添加**-f**参数，Docker会发送**SIGKILL信号**给容器