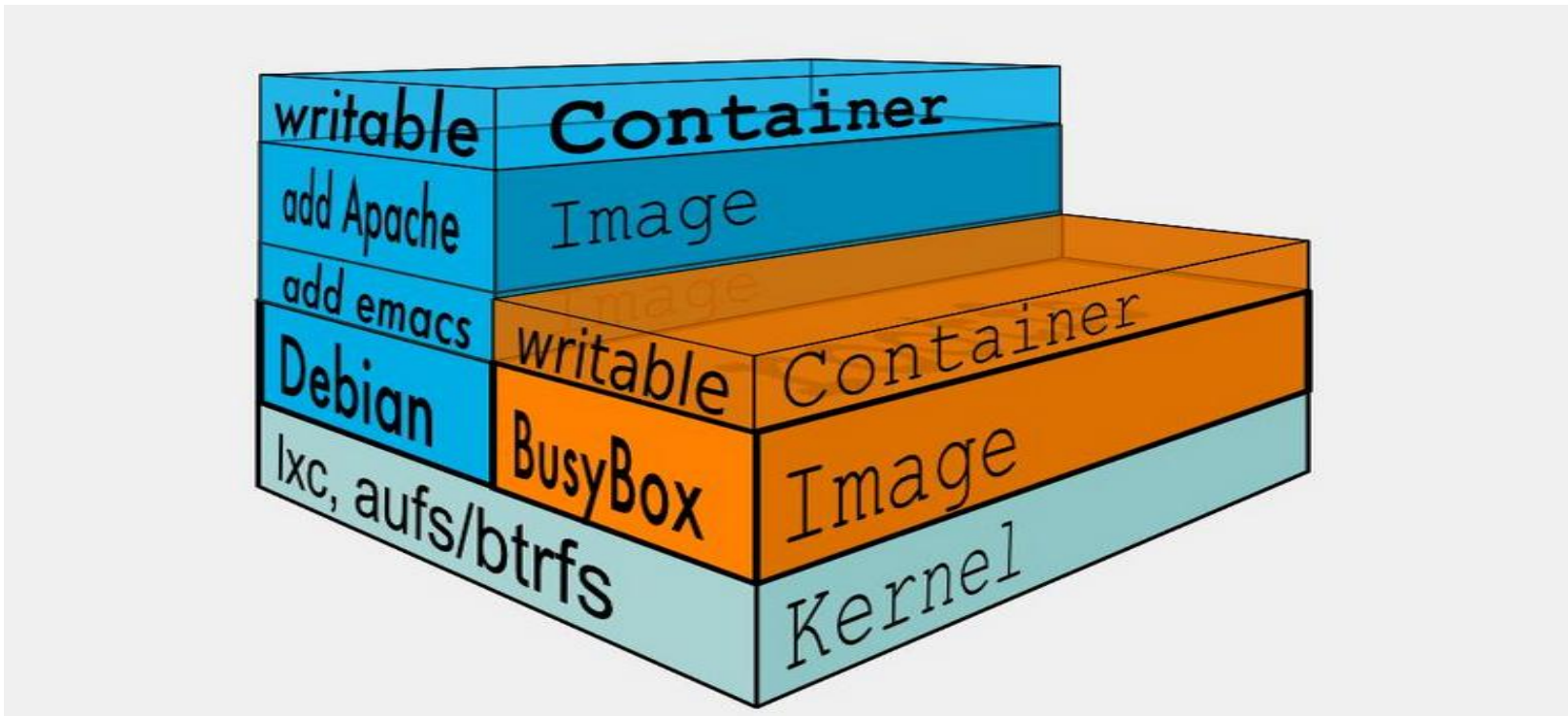
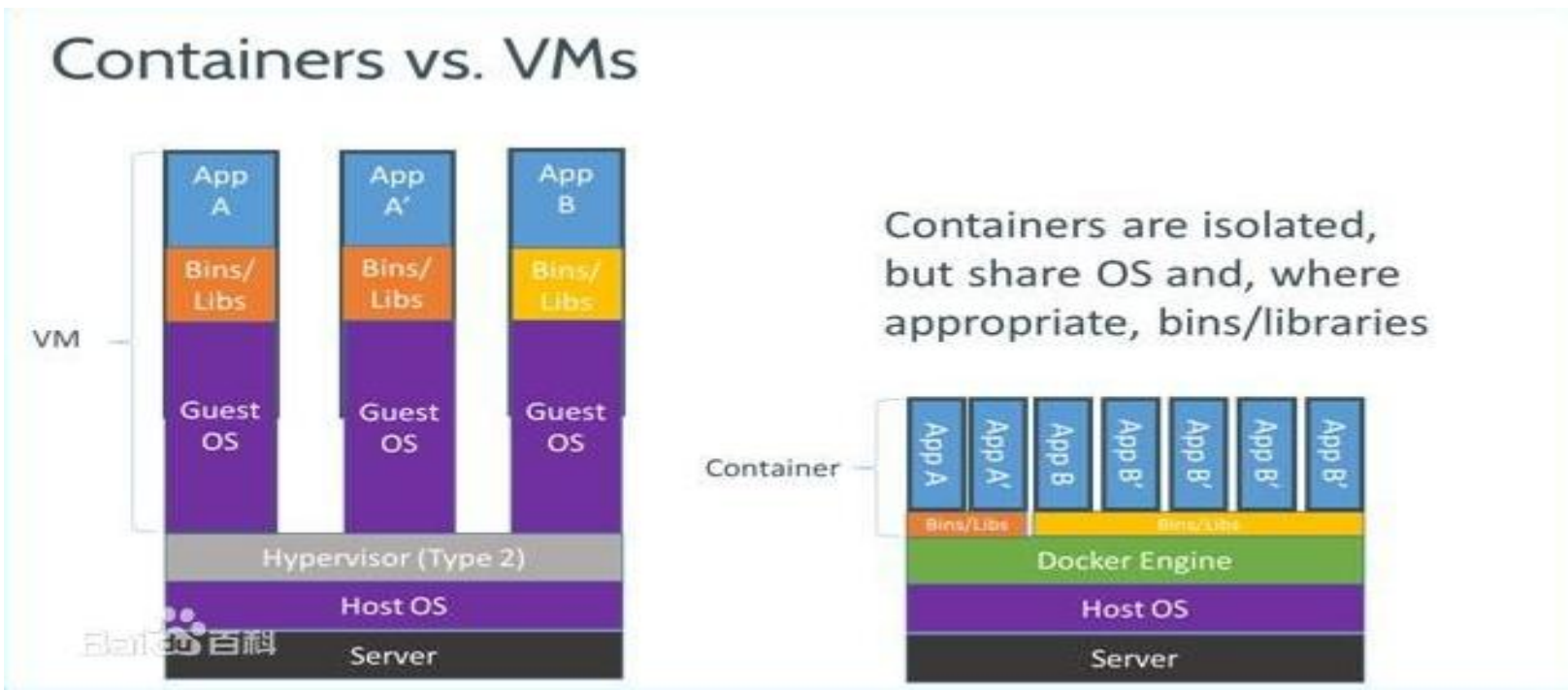
Docker 的介绍：

Docker 是一个基于 lxc、cgroup、namespace 的开源应用容器引擎，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，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），相对来说开销比较小。



与传统虚拟化的比较：



1） 它的启动速度快

2） 与宿主机共用同一个操作系统内核

3） 系统的开销比较小

docker 的安装

```
sudo apt-get update

sudo apt-get install -y docker.io
```

docker 服务的启动：

```
sudo service docker.io start

sudo service docker.io status
```

docker 镜像的下载：

```
sudo docker pull ubuntu
```

下载指定版本的镜像:

sudo docker pull ubuntu:14.04

sudo docker pull centos:7.1

docker 镜像信息的查看:

sudo docker images

使用 docker tag 为本地镜像添加新的标签:

sudo docker tag dl.dockerpool.com:5000/centos7.1

使用 docker inspect 获取该镜像的详细信息:

sudo docker inspect f1dade627e25

使用 docker inspect 中的-f 参数来指定显示其中一项内容时:

sudo docker inspect -f {{".Architecture"}} f1dade627e25

amd64

使用 docker search 搜索镜像

--autumated=false 仅显示自动创建的镜像

--no-trunc=false 输出信息不截断显示

-s,--stars=0 指定仅显示评价为指定星级以上的镜像

例如:

sudo docker search mysql

使用 docker rmi 删除镜像:

删除一个 image,首先停止它上面的 container，然后删除这些 continer，然后执行下面的操作

sudo docker stop 运行的 container 的 id

sudo docker rm container 的 id

sudo docker rmi ubuntu

sudo docker rmi -f ubuntu -f 参数为强制删除（这种不推荐使用）

使用 docker commit 基于已有镜像容器的创建:

格式为：docker commit [OPTION] CONTAINER [REPOSITORY]

主要包含选项：

-a,--author="" 作者信息

-m,--messages=""提交消息

-p,--pause=true 提交时暂停容器运行

我们首先启动一个镜像，并在其中进行操作，例如创建一个 test 文件，之后退出记下 ID:

```
root@docker:~# sudo docker run -ti ubuntu:14.04 /bin/bash
root@6023e4e4daf4:/# touch test
root@6023e4e4daf4:/# exit
exit
```

此时该容器已经发生了改变，可以使用 docker commit 命令提交为一个新的镜像，提交时可以使用 id 或名称来指定容器：

```
root@docker:~# sudo docker commit -m "Added a new file" -a "Docker Newbee" 6023e4e4daf4 test
2e11586b4fa1eee75d2658c91ceffa35344e8f7e2619ffe9fd46325913652598
root@docker:~# sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL
SIZE
test                latest              2e11586b4fa1        41 seconds ago      188.3 MB
```

docker run 常用的选项有：

-e 设置容器的运行 env 环境变量

-v 映射服务的一个目录到容器中

-p 容器对服务器暴露的端口

-c cpu 使用的权重

-m 限制容器的内存使用量

-i 标准输出到当前 term

-t 分配一个 tty

基于本地模板的导入：

我们可以使用 OPENVZ 提供的模板来创建。下载地址：

https://download.openvz.org/template/precreated/

这里，我下载的是 centos-7-x86_64-minimal.tar.gz 这个压缩包，可以使用下面命令导入：

```
sudo cat centos-7-x86_64-minimal.tar.gz | docker import - centos:7
```

存出和载入镜像：

存出镜像：使用 docker save 命令

sudo docker save -o ubuntu_14.04.tar ubuntu:14.04

载入镜像：可以使用 docker load 从存出的本地文件中再导入到本地镜像库

sudo docker load --input ubuntu_14.04.tar

或

sudo docker load < ubuntu_14.04.tar

上传镜像：

默认上传到 DockerHub 官方仓库（需要登录），命令格式为：docker push NAME[:TAG]

用户在 DockerHub 网站上注册后，即可上传自制的镜像。例如 user 上传本地的 test:latest 镜像，

可以先添加新的标签 user/test:latest,然后用 docker push 命令上传镜像：

sudo docker tag test:latest user/test:latest

sudo docker push user/test:latest

....

....

username:

password:

Email:xxx@xxx.com

第一次使用时，会提示输入登录信息或进行注册

如何通过 ssh 方式进入，具体做法：

wget 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

cp nsenter /usr/local/bin/

http://jpetazzo.github.io/2014/06/23/docker-ssh-considered-evil/

docker run -v /usr/local/bin:/target jpetazzo/nsenter

会自动安装 nsenter 这个工具

然后登陆到容器：

docker inspect --format "{{.State.Pid}}" 050fe3262753 #获取容器的 pid

sudo nsenter --target 3190 --mount --uts --ipc --net --pid

sudo docker-enter ID/name

```
#!/bin/sh

if [ -e $(dirname "$0")/nsenter ]; then

   NSENTER=$(dirname "$0")/nsenter

else

  NSENTER=nsenter

fi

if [ -z "$1" ]; then

  echo "Usage: `basename "$0"` CONTAINER [COMMAND [ARG]...]"

else

  PID=$(docker inspect --format "{{.State.Pid}}" "$1")

  if [ -z "$PID" ]; then

    exit 1

  fi

  shift

  OPTS="--target $PID --mount --uts --ipc --net --pid --"

  if [ -z "$1" ]; then

    "$NSENTER" $OPTS su - root

  else

      "$NSENTER" $OPTS env --ignore-environment -- "$@"

  fi

fi
```

进入后开启服务，及在宿主机上 iptables 的设置

iptables -t nat -A PREROUTING -m tcp -p tcp --dport 9100 -j DNAT --to-destination 192.168.2.2:80

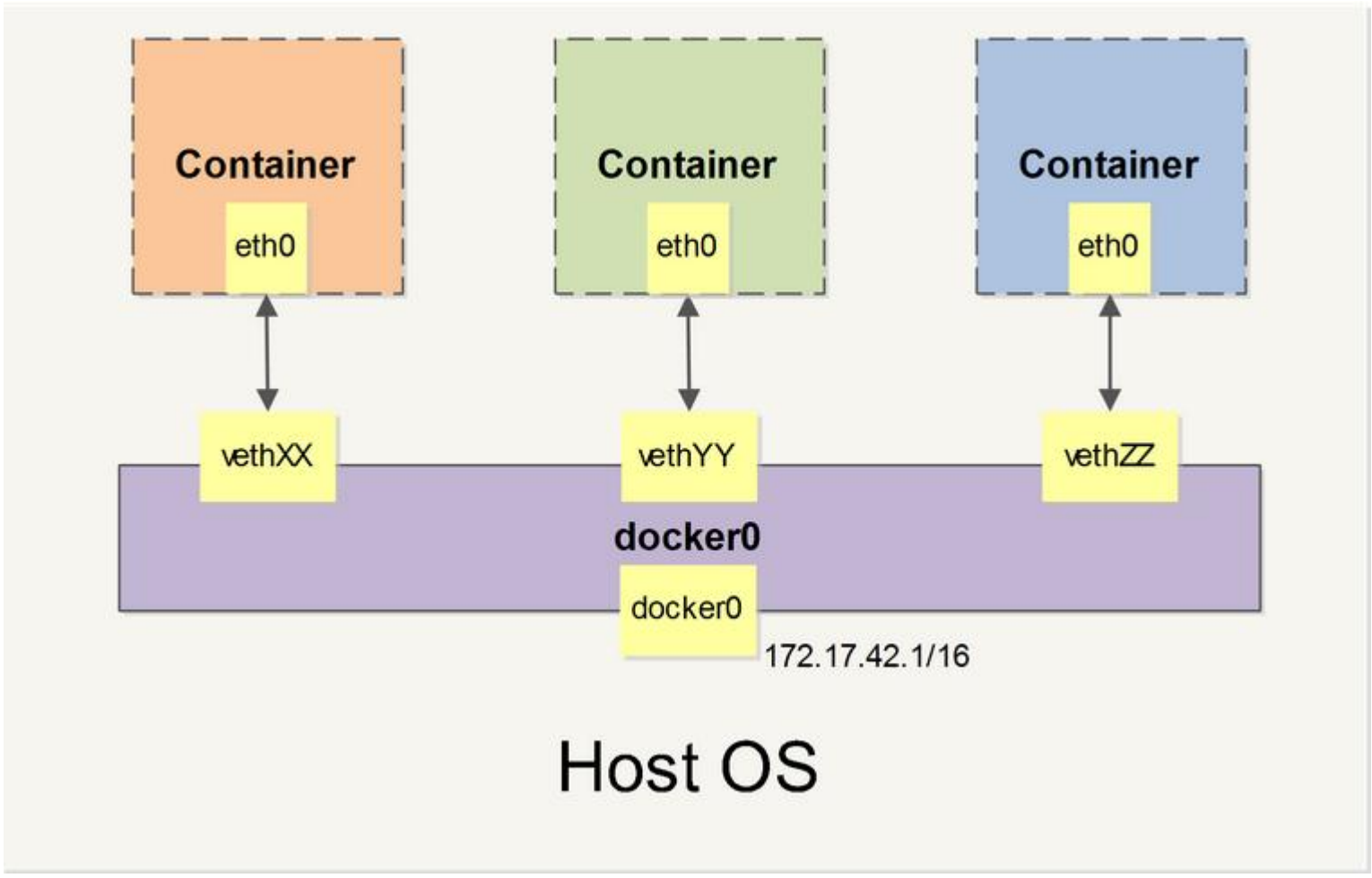
iptables -t nat -A POSTROUTING -m tcp -p tcp --dport 80 -d 192.168.2.2 -j SNAT --to-source 10.1.16.140

iptables -A INPUT -p tcp -m tcp -dport 9100 -j ACCEPT

```
docker run -it -d -name=zabbix -p 9100:80  centos:6  sh
```

自定义网桥

默认的 docker0 的网络模式：



除了默认的 docker0 网桥，用户也可以指定网桥来连接各个容器。

在启动 Docker 服务的时候，使用 -b BRIDGE 或 --bridge=BRIDGE 来指定使用的网桥。

如果服务已经运行，那需要先停止服务，并删除旧的网桥。

```
$ sudo service docker stop

$ sudo ip link set dev docker0 down

$ sudo brctl delbr docker0
```

然后创建一个网桥 bridge0

```
$ sudo brctl addbr bridge0

$ sudo ip addr add 192.168.5.1/24 dev bridge0

$ sudo ip link set dev bridge0 up
```

查看确认网桥创建并启动。

```
$ ip addr show bridge0
```

配置 Docker 服务，默认桥接到创建的网桥上。

```
$ echo 'DOCKER_OPTS="-b=bridge0"' >> /etc/default/docker
```

$ sudo service docker start

docker0.sh

```
stop docker
ip link set dev docker0 down
brctl delbr docker0
brctl addbr docker0
ip addr add 192.168.2.1/24 dev docker0
ip link set dev docker0 up
echo 'DOCKER_OPTS="-b=bridge0"' >> /etc/default/docker
start docker
```

sh docker0.sh