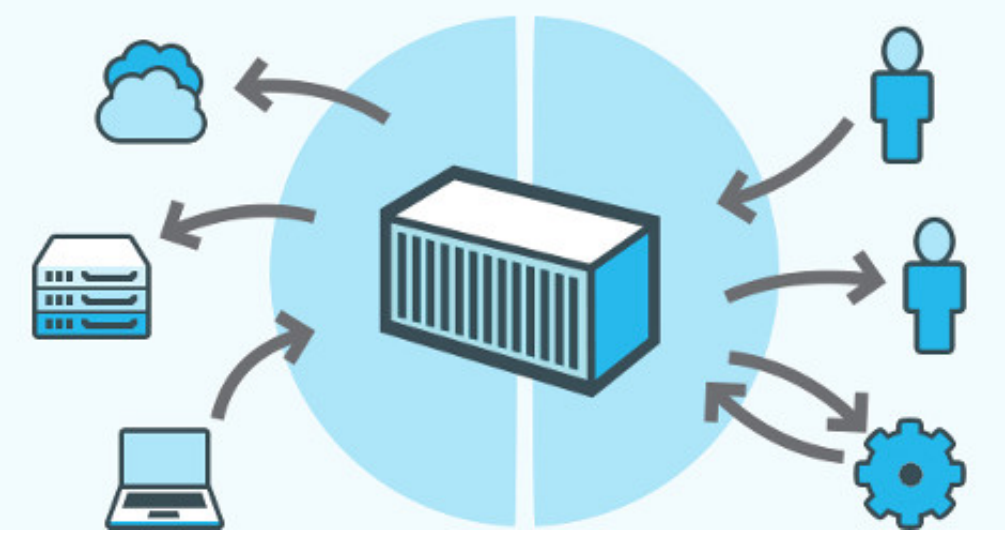
Docker Internal Technical Sharing

Docker

Introduction to Docker

What is Docker

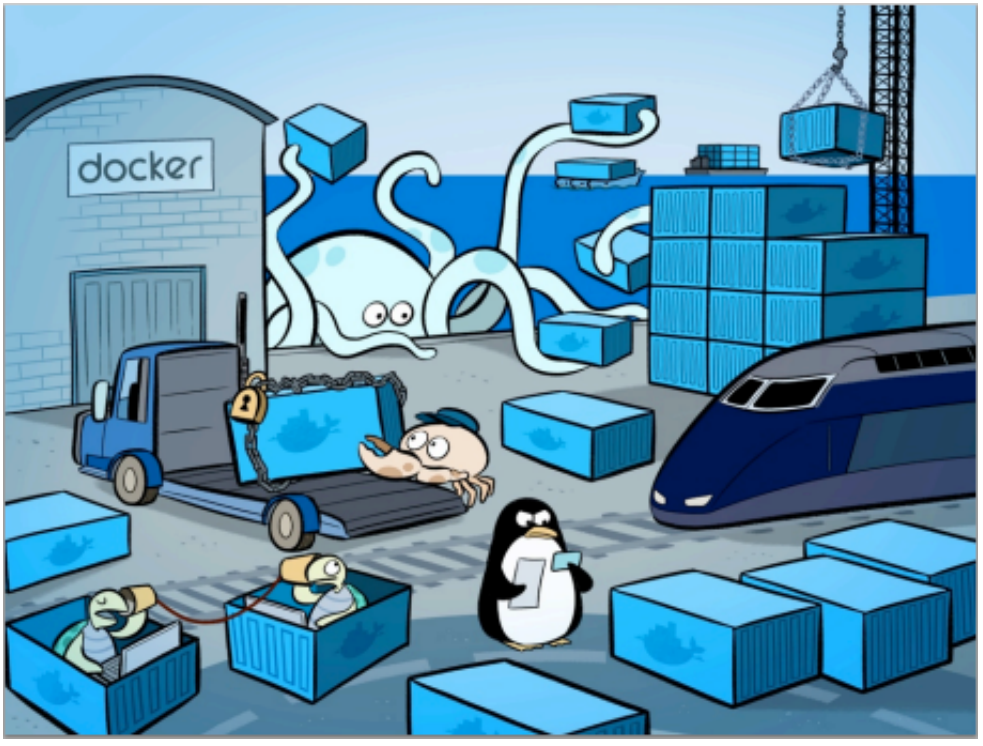
Docker - An open platform for developers and sysadmins to build, ship, and run distributed applications



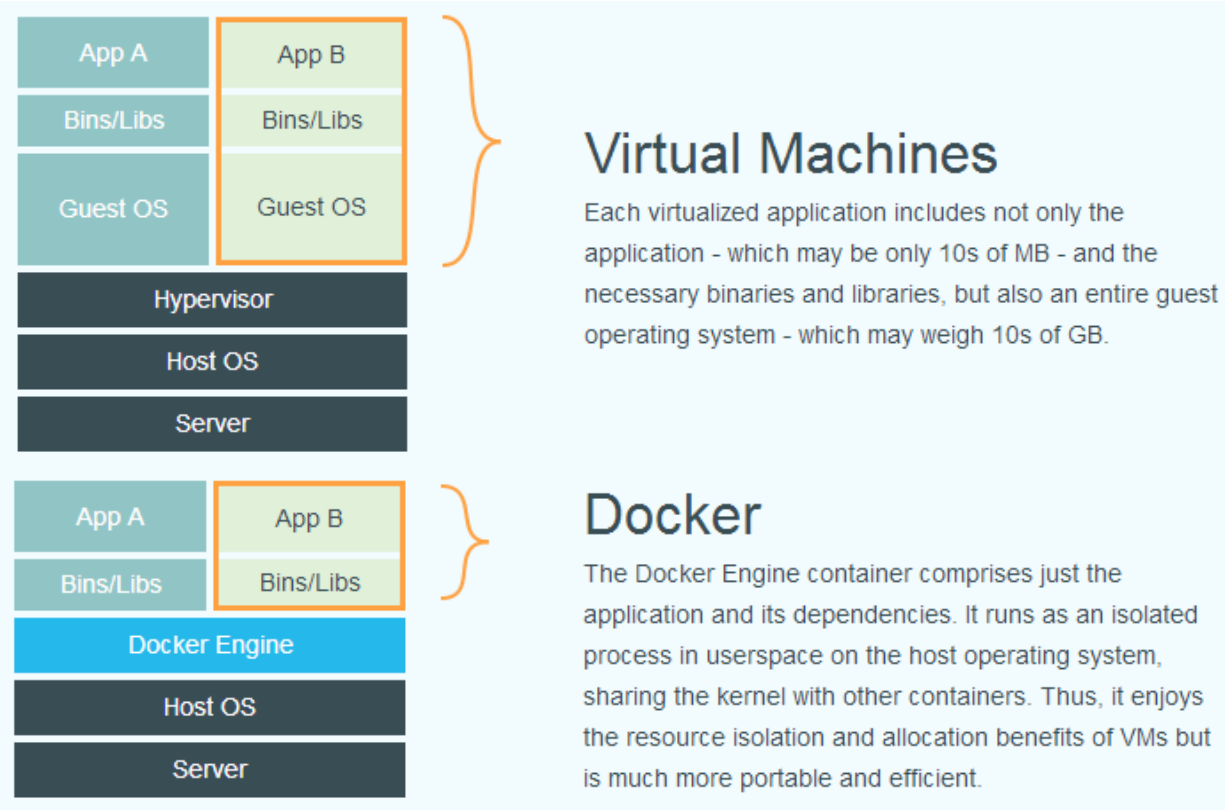
Why we need Docker

- Developers
  - Build any app in any language using any toolchain
  - Get going quickly by starting with one of the 13,000+ apps available on Docker Hub
  - Helps developers build and ship higher-quality applications, faster.

- Sysadmins
  - Provide standardized environments for their development, QA, and production teams, reducing “works on my machine” finger-pointing



Difference from VM



特性	容器	虚拟机
启动	秒级	分钟级
硬盘使用	一般为 MB	一般为 GB
性能	接近原生	弱于
系统支持量	单机支持上千个容器	一般几十个

Basic Concepts

Image

Docker 镜像就是一个只读的模板,可以用来创建 Docker 容器
相当于*VMware*里的*template*，可以安装好相应的应用，设置好，供其他人*clone*后使用

Container

Docker 利用容器来运行应用。
容器是从镜像创建的运行实例。它可以被启动、开始、停止、删除。每个容器都是相互隔离的、保证安全的平台。
可以把容器看做是一个简易版的 Linux 环境（包括root用户权限、进程空间、用户空间和网络空间等）和运行在其中的应用程序。
相当于*Vmware*里的*VM*

Repository

存放Image的地方
自动创建

数据管理

Data volumes

简单来说，Volume就是目录或者文件，它可以绕过默认的UFS，而以正常的文件或者目录的形式存在于宿主机上，可以提供很多有趣的特性：

- 数据卷可以在容器之间共享和重用
- 对数据卷的修改会立马生效
- 对数据卷的更新，不会影响镜像
- 卷会一直存在，直到没有容器使用
  类似于 *Linux* 下对目录或文件进行 *mount, OpenStack*上也有类似的东西

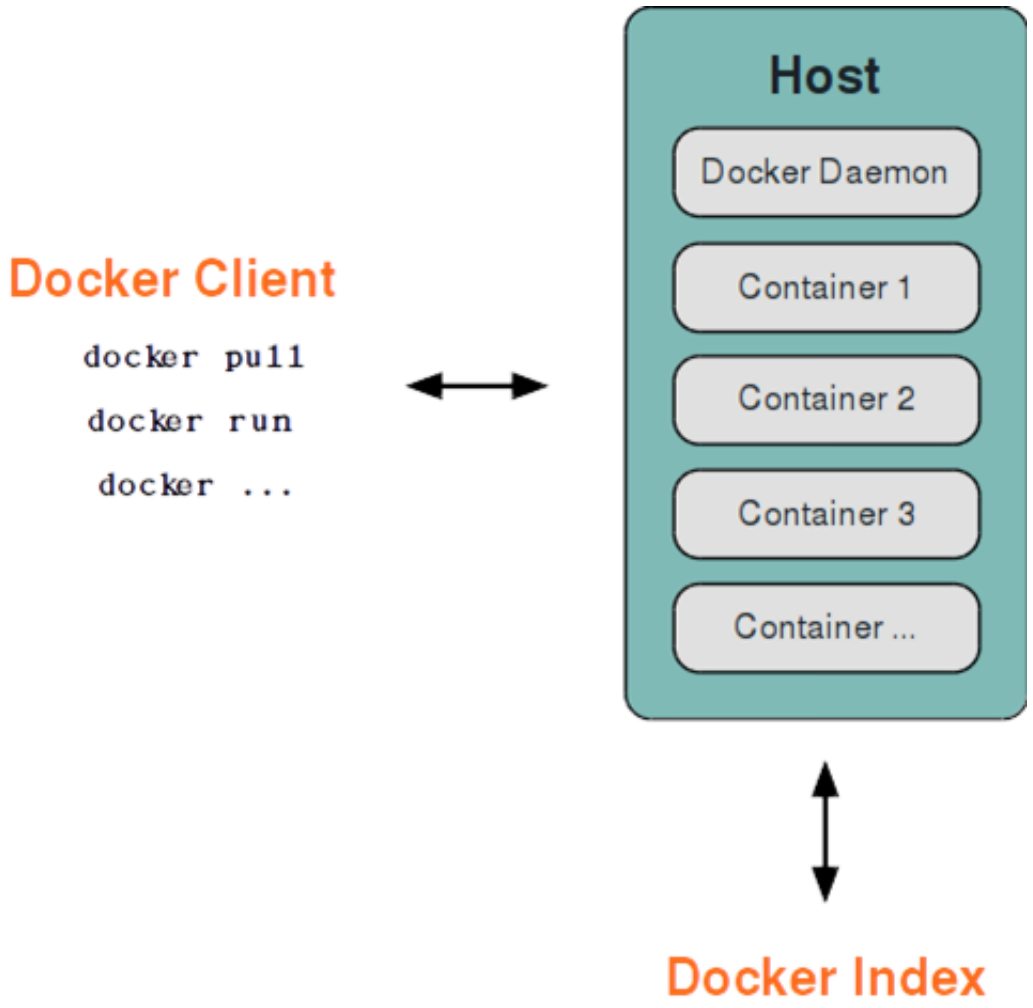
Data volume containers

数据卷容器，其实就是一个正常的容器，专门用来提供数据卷供其它容器挂载的。常见的使用场景是使用纯数据容器来持久化数据库、配置文件或者数据文件等

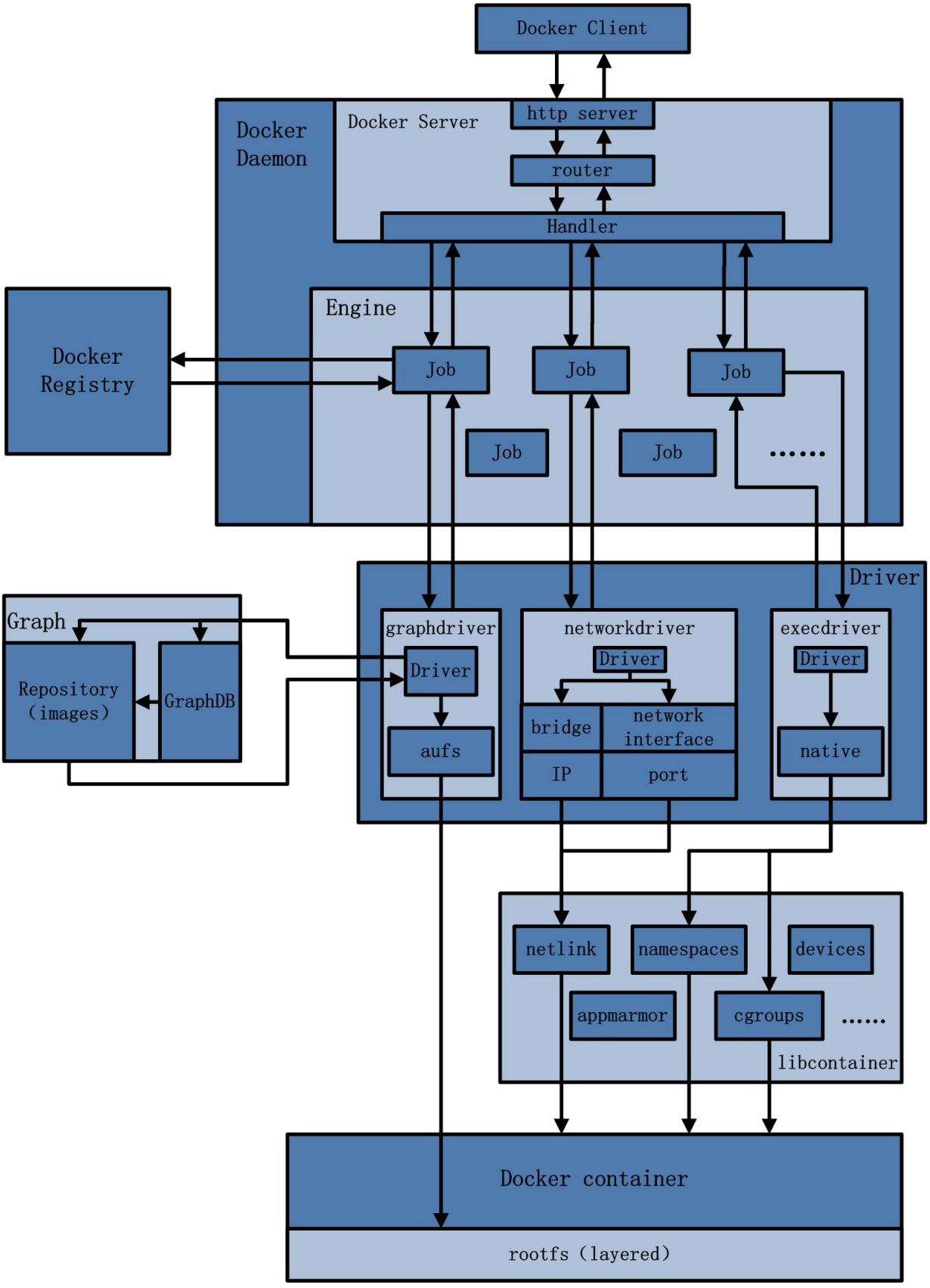
实现原理

基本架构

简单版


Host
Docker Daemon
Container 1
Container 2
Container 3
Container ...
Docker Client
docker pull
docker run
docker ...
Docker Index

复杂版


Docker Client
Docker Daemon
Docker Server
http server
router
Handler
Engine
Docker Registry
Job
Job
Job
Job
Job
.....
Driver
Graph
Repository (images)
GraphDB
graphdriver
Driver
aufs
networkdriver
Driver
bridge
network interface
IP
port
execdriver
Driver
native
netlink
namespaces
devices
appmarmor
cgroups
.....
libcontainer
Docker container
rootfs (layered)

命名空间

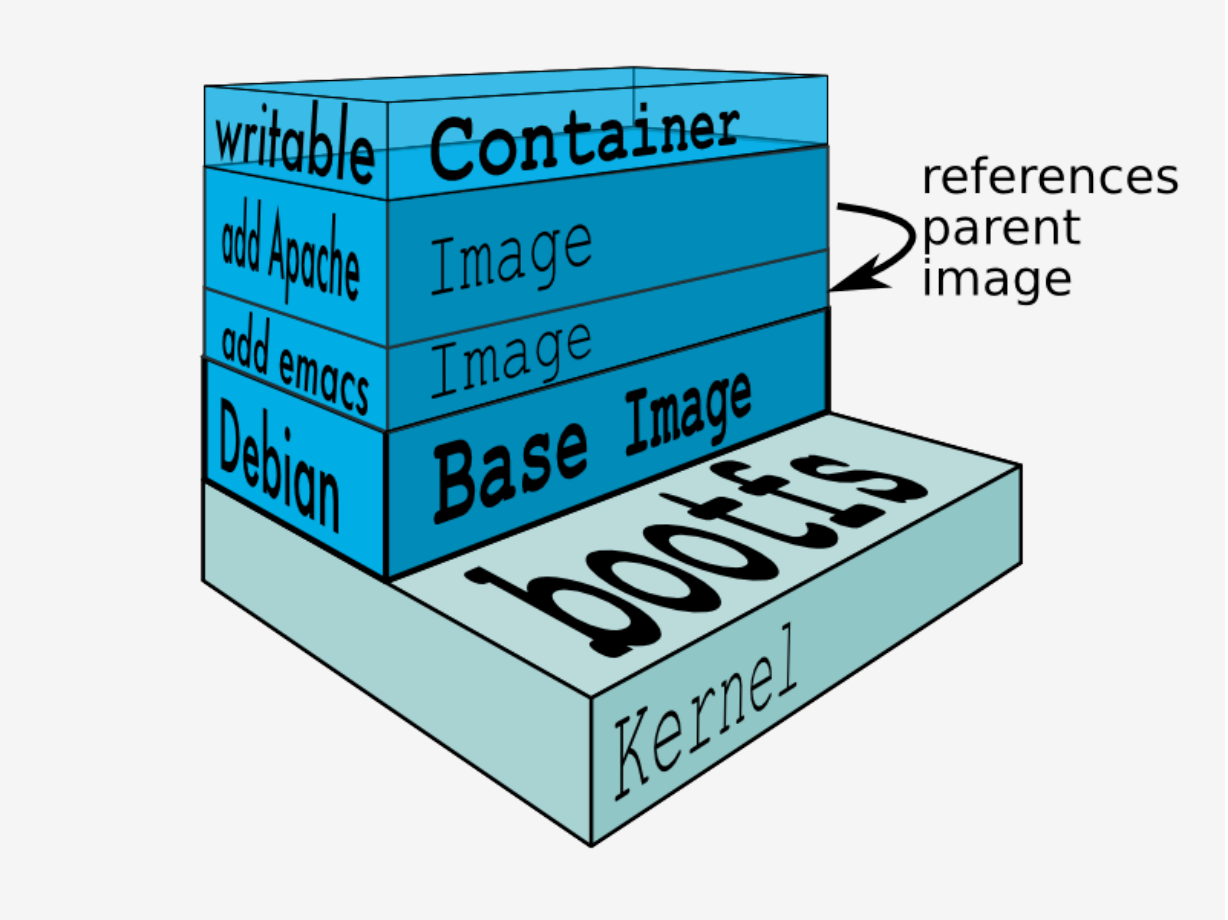
Docker takes advantage of a technology called namespaces to provide the isolated workspace we call the container. When you run a container, Docker creates a set of namespaces for that container.
This provides a layer of isolation: each aspect of a container runs in its own namespace and does not have access outside it.
Some of the namespaces that Docker uses are:
- The pid namespace: Used for process isolation (PID: Process ID).
- The net namespace: Used for managing network interfaces (NET: Networking).
- The ipc namespace: Used for managing access to IPC resources (IPC: InterProcess Communication).
- The mnt namespace: Used for managing mount-points (MNT: Mount).
- The uts namespace: Used for isolating kernel and version identifiers. (UTS: Unix Timesharing

Control groups

Docker also makes use of another technology called cgroups or control groups. A key to running applications in isolation is to have them only use the resources you want. This ensures containers are good multi-tenant citizens on a host. Control groups allow Docker to share available hardware resources to containers and, if required, set up limits and constraints. For example, limiting the memory available to a specific container.

Union FS

每个镜像都由很多层次构成，Docker 使用 Union FS 将这些不同的层结合到一个镜像中去。
Union file systems, or UnionFS, are file systems that operate by creating layers, making them very lightweight and fast. Docker uses union file systems to provide the building blocks for containers.



从一个image启动一个container，Docker会先加载这个image和依赖的父images以及base image，用户的进程运行在writeable的layer中。所有parent image中的数据信息以及 ID、网络和lxc管理的资源限制等具体container的配置，构成一个Docker概念上的container

Container format

Docker combines these components into a wrapper we call a container format. The default container format is called libcontainer. Docker also supports traditional Linux containers using LXC. In the future, Docker may support other container formats, for example, by integrating with BSD Jails or Solaris Zones.

Reference

官网介绍
Docker——从入门到实践

Try it

https://www.docker.com/tryit/

Installation

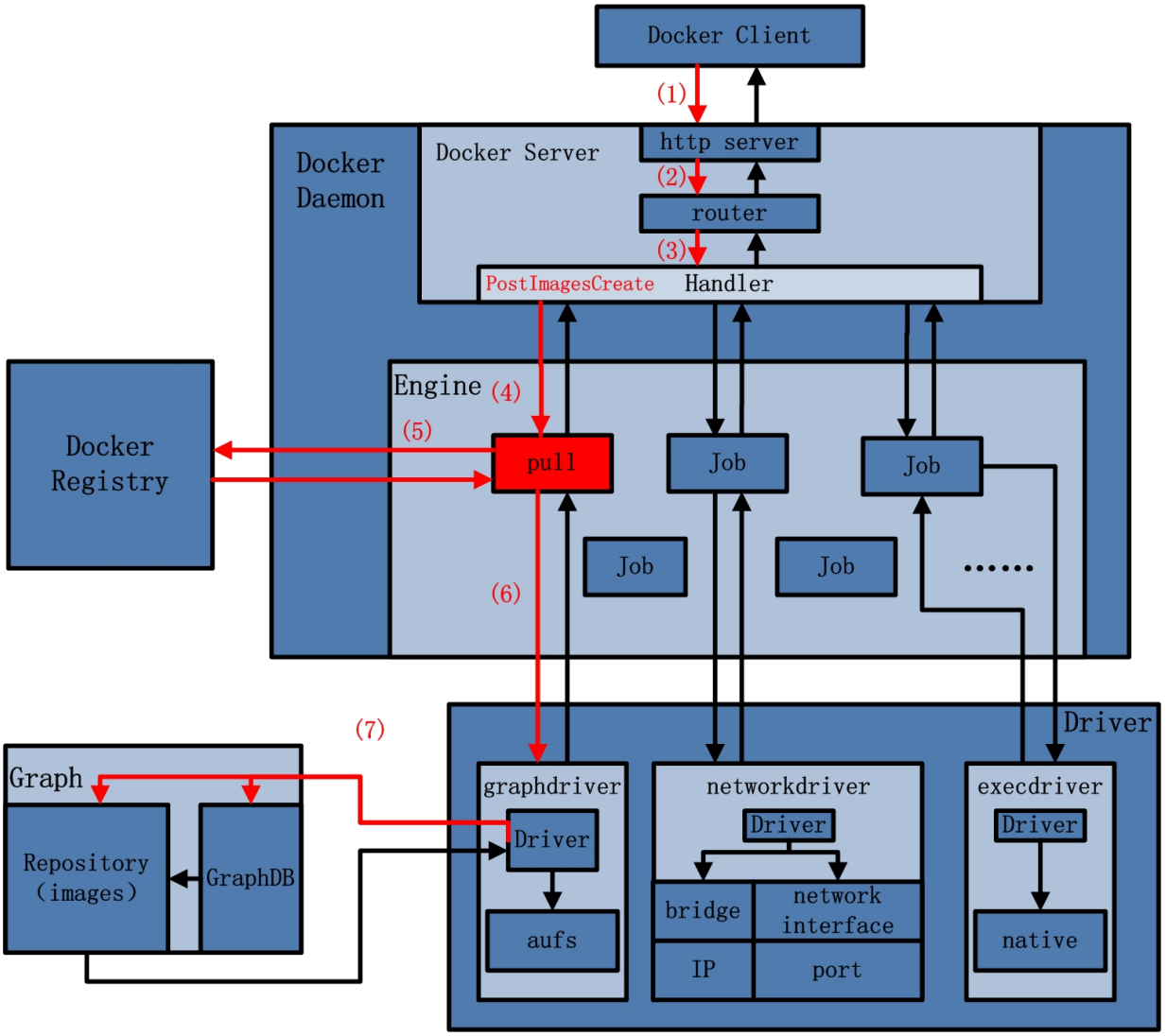
https://docs.docker.com/installation

```
- rpm -iUvh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
- yum update -y
- yum -y install docker-io
- service docker start
- chkconfig docker on
```

Download images

https://registry.hub.docker.com/

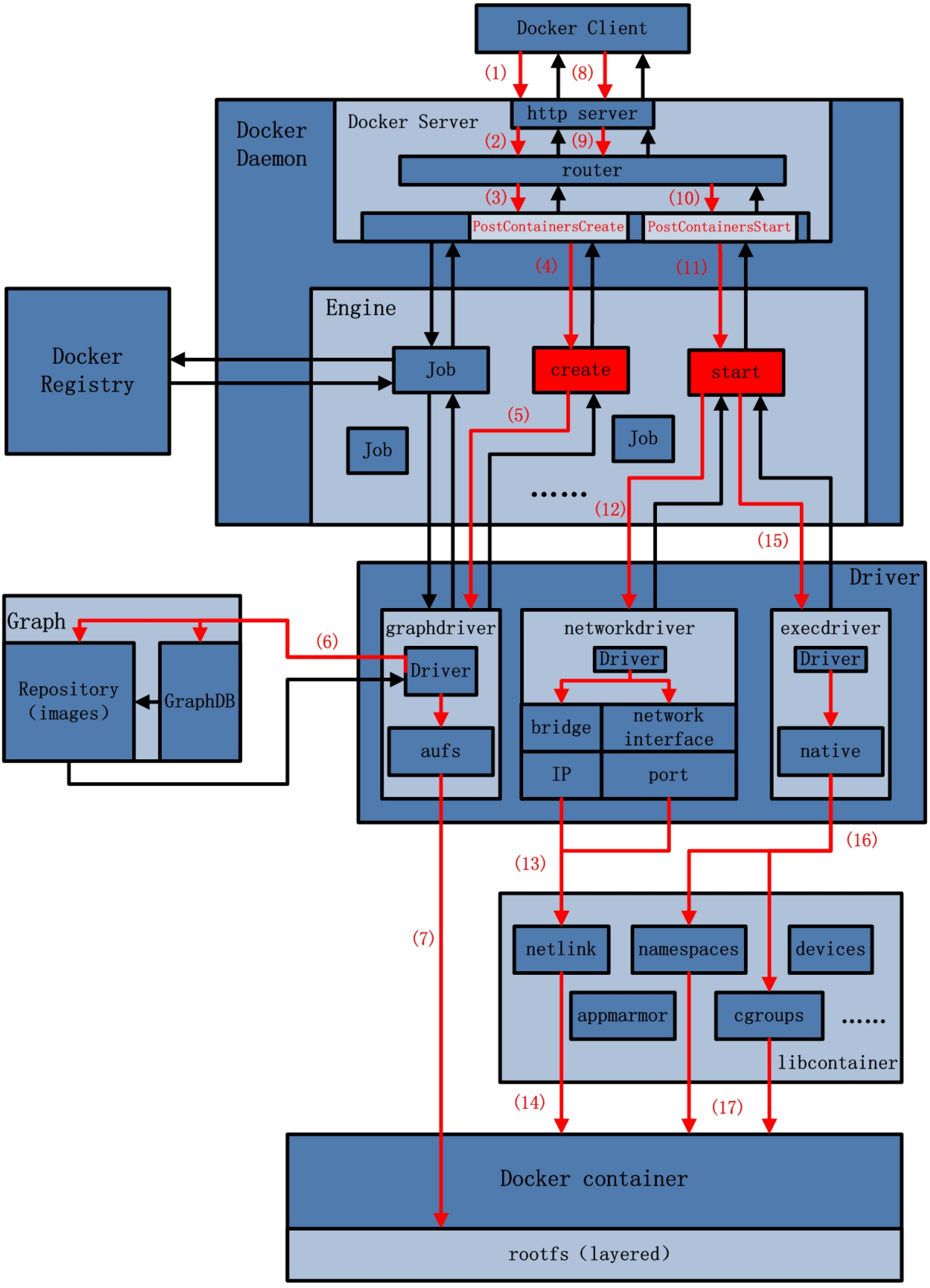
```
docker pull centos
docker pull ubuntu
docker images
```



Run a container

Hello world

```
docker run ubuntu:14.04 /bin/echo 'Hello world'
```



启动一个 bash 终端，允许用户进行交互

```
docker run -t -i ubuntu:14.04 /bin/bash
```

-t 选项让*Docker*分配一个伪终端（ *pseudo-tty* ）并绑定到容器的标准输入上， *-i* 则让容器的标准输入保持打开

守护态运行

```
ID=$( docker run -d centos /usr/bin/top -b)
docker attach $ID
docker stop $ID
```

启动已终止容器
可以利用 docker start 命令，直接将一个已经终止的容器启动运行。“`

Create new image

from an old image

```
sudo docker commit -m "PHP installed" 0b2616b0e5a8 emma:v2
```

from dockerfile

```
Dockfile
# This is a comment
FROM ubuntu:14.04
MAINTAINER Docker Newbee <newbee@docker.com>
RUN apt-get -qq update
RUN apt-get -qqy install ruby ruby-dev
RUN gem install sinatra
```

生成image

```
docker build -t="emma:v3"
```

Import openvz templates

http://openvz.org/Download/templates/precreated

```
cat ubuntu-14.04-x86_64-minimal.tar.gz  |docker import -
ubuntu:14.04
```