引言

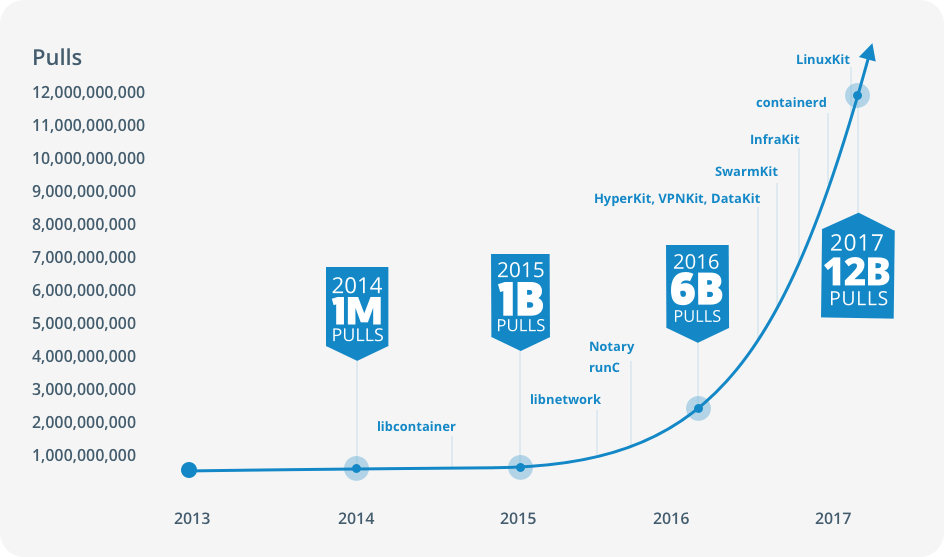
Docker 自2013年以来非常火热，无论是从 Github 上的代码活跃度，还是 Redhat 在 RHEL6.5 中集成对 Docker 的支持, 就连 Google 的 Compute Engine 也支持 Docker。京东 618，使用了 15 万个 Docker 实例，并将所有业务全部容器化。腾讯和阿里也早在 2015 年就实现了万台 Docker 应用实践。归根结底还是依赖于 Docker 的简单部署，解放运维。

如果您还在用常规方式去部署安装 PHP？Tomocat？MySQL？WordPress？那你就 Out 了。赶快来一起学习吧！

Docker 是什么

Docker 最初是 dotCloud 公司创始人 Solomon Hykes 在法国期间发起的一个公司内部项目，它是基于 dotCloud 公司多年云服务技术的一次革新，并于 2013 年 3 月以 Apache 2.0 授权协议开源，主要项目代码在 GitHub 上进行维护。Docker 项目后来还加入了 Linux 基金会，并成立推动 开放容器联盟（OCI）。

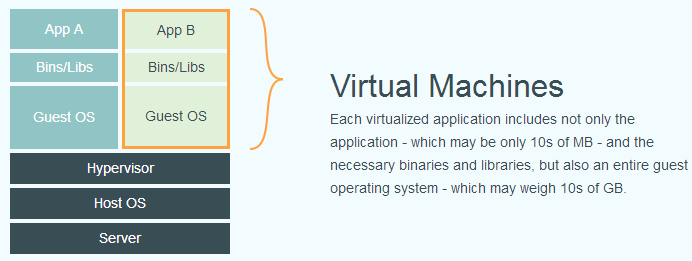
Docker 自开源后受到广泛的关注和讨论，至今其 GitHub 项目已经超过 4 万九千个星标和一万多个 fork。甚至由于 Docker 项目的火爆，在 2013 年底，dotCloud 公司决定改名为 Docker。Docker 最初是在 Ubuntu 12.04 上开发实现的；Red Hat 则从 RHEL 6.5 开始对 Docker 进行支持；Google 也在其 PaaS 产品中广泛应用 Docker。

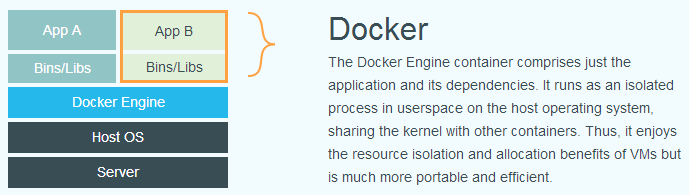


Docker 使用 Google 公司推出的 Go 语言 进行开发实现，基于 Linux 内核的 cgroup，namespace，以及 AUFS 类的 Union FS 等技术，对进程进行封装隔离，属于操作系统层面的虚拟化技术。由于隔离的进程独立于宿主和其它的隔离的进程，因此也称其为容器。最初实现是基于 LXC，从 0.7 版本以后开始去除 LXC，转而使用自行开发的 libcontainer，从 1.11 开始，则进一步演进为使用 runC 和 containerd。

Docker 在容器的基础上，进行了进一步的封装，从文件系统、网络互联到进程隔离等等，极大的简化了容器的创建和维护。使得 Docker 技术比虚拟机技术更为轻便、快捷。

下面的图片比较了 Docker 和传统虚拟化方式的不同之处。传统虚拟机技术是虚拟出一套硬件后，在其上运行一个完整操作系统，在该系统上再运行所需应用进程；而容器内的应用进程直接运行于宿主的内核，容器内没有自己的内核，而且也没有进行硬件虚拟。因此容器要比传统虚拟机更为轻便。







Docker 思想

传统的方式中，在服务器中搭建一个常见的 LAMP（Linux＋Apache＋Mysql＋PHP）网站的步骤：

* 下载安装 Apache，Mysql 和 PHP 以及其它的相关依赖环境。
* 分别进行配置和测试。
* 整合到一起进行功能测试。

这种方式操作枯燥繁琐，将不同的软件集成起来的过程中有很多不可控的风险，最无法忍受的是，一旦需要重新迁移服务器或者重新部署一套环境，这样的琐碎和无趣的“体力活”还将重新执行一遍。

针对这个问题，以前我们通常用虚拟机集成部署好一套环境，然后做成一个虚拟机模版来解决。不过这种方式有资源利用率低，灵活性差和迁移平台限制等问题。例如，我们想要的仅仅是一个 LAMP 环境，但是用这种方式却必须集成打包一个完整的操作系统（每个虚拟机需要单独分配独占的内存,磁盘等资源）。此外，这样打包后的一套 LAMP 环境中，软件之间版本依赖性强，很难进行变更更新。最后，通常这样的虚拟机还需要绑定特定的虚拟化管理程序来管理，这样对于迁移的平台就有了限制。

如今，Docker 通过对 Linux 容器技术 LXC（Linux Containers）等进一步优化，提供了各种容器管理工具，通过容器来管理应用，这样便可以达到 “ **Build Once, Works Everywhere**” 的目的。也就是说对于开发和部署来说，使用 Docker 可以：

* 更快速的交付和部署应用环境。
* 更高效的资源利用率。
* 更便捷的迁移和扩展性。
* 更便捷的应用更新管理。

那么我们接来下说一下 Docker 核心技术。

Docker 镜像、容器、仓库

**镜像**

docker 镜像是一个特殊的文件系统，除了提供容器运行时所需的程序、库、资源、配置等文件外，还包含了一些为运行时准备的一些配置参数（如匿名卷、环境变量、用户等）。镜像不包含任何动态数据，其内容在构建之后也不会被改变。

**容器**

镜像（Image）和容器（Container）的关系，就像是面向对象程序设计中的 类 和 实例 一样，镜像是静态的定义，容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等。

容器的实质是进程，但与直接在宿主执行的进程不同，容器进程运行于属于自己的独立的 命名空间。因此容器可以拥有自己的 root 文件系统、自己的网络配置、自己的进程空间，甚至自己的用户 ID 空间。容器内的进程是运行在一个隔离的环境里，使用起来，就好像是在一个独立于宿主的系统下操作一样。这种特性使得容器封装的应用比直接在宿主运行更加安全。

**仓库**

镜像构建完成后，可以很容易的在当前宿主机上运行，但是，如果需要在其它服务器上使用这个镜像，我们就需要一个集中的存储、分发镜像的服务，Docker Registry 就是这样的服务。

一个 Docker Registry 中可以包含多个仓库（Repository）；每个仓库可以包含多个标签（Tag）；每个标签对应一个镜像。

通常，一个仓库会包含同一个软件不同版本的镜像，而标签就常用于对应该软件的各个版本。我们可以通过 <仓库名>:<标签> 的格式来指定具体是这个软件哪个版本的镜像。如果不给出标签，将以 latest 作为默认标签。

Docker 的安装

这边仅以 Linux Centos7 为例：

Docker 要求 CentOS 系统的内核版本高于 3.10 ，查看本页面的前提条件来验证你的 CentOS 版本是否支持。

通过 uname - r 命令查看你当前的内核版本。

[root@centos ~]# uname -r ==》查看内核版本

[root@centos ~]# yum -y install docker ==》 安装docker

[root@centos ~]# service docker start ==》 启动docker服务

**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，我使用的是网易的镜像地址：http://hub-mirror.c.163.com。

新版的 Docker 使用 /etc/docker/daemon.json（Linux） 来配置 Daemon。

在该配置文件中加入（没有该文件的话，请先建一个）：

{

"registry-mirrors": ["https://c.163.com/hub#/m/home/"]

}

Docker 网络

**docker 网络具备四种网络模式**：

* bridge 模式，使用 –net=bridge 指定，默认设置。
* none 模式， 使用 –net=none 指定。
* host 模式，使用 –net=host 指定。
* container 模式，使用 –net=container: 容器名称或 ID 指定

**bridge 模式**

docker 网络隔离基于网络命名空间，在物理机上创建 docker 容器时会为每一个 docker 容器分配网络命名空间，并且把容器IP桥接到物理机的虚拟网桥上。

**none模式**

故名思议，none 网络就是什么都没有的网络。挂在这个网络下的容器除了 lo，没有其他任何网卡。

**host模式**

连接到 host 网络的容器共享 docker host 的网络栈，容器的网络配置与 host 完全一样。

**container 模式**

此模式和 host 模式很类似，只是此模式创建容器共享的是其他容器的 IP 和端口而不是物理机，此模式容器自身是不会配置网络和端口，创建此模式容器进去后，你会发现里边的 IP 是你所指定的那个容器 IP 并且端口也是共享的，而且其它还是互相隔离的，如进程等。

**docker 网络初始化的过程如下**：

初始化设备 —> 初始化 bridge —> 初始化 iptables —> 初始化 ip foward(内核路由转发) — > register network job function —> end

如果需要更加详细了解 Docker 的网络模型可以参考这篇文章：[docker 四种网络模型](https://blog.csdn.net/csdn066/article/details/77165269)，我们这边就不再累述。

Docker Hello World

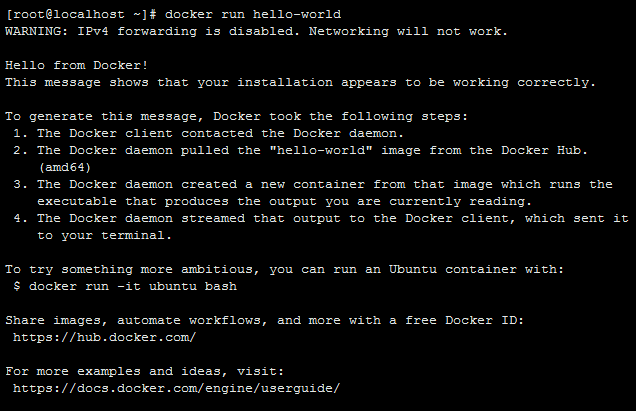
上文中 Docker 环境，我们已经搭建完毕，下面我们来运行一个示例看看我们得环境到底正不正确。我们知道在学习各种语言时候，我们首先学会的就是如何运行 hello，world。

通过他，我们来认识这个世界。

docker 的 hello，world 其实很简单，使用 docker 指令 直接从远端拉取镜像运行：

具体指令：  
[root@centos ~]# docker run hello-world

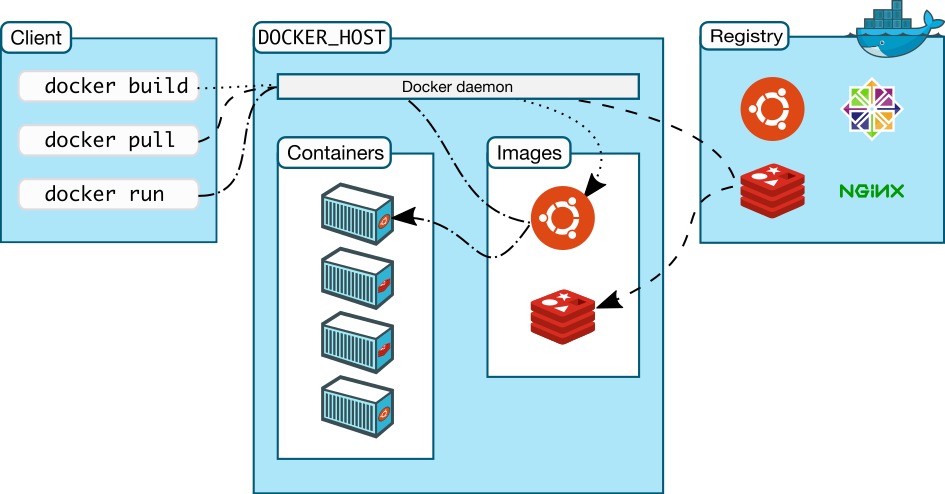
我们先不管 为什么镜像名称为 hello-world，我们先看运行效果：如果出现了以下画面，则可以认为我们得 Docker 环境搭建成功。



好了，我们来分析一下，运行 hello-world 之后的打印信息。

* 首先我们看到 "Hello from Docker!"，说明我们之前的配置没有问题。
* 为了产生这些信息，docker 经过了这些步骤
  + docker 客户端连接 docker daemon 进程。
  + docker daemon 向 docker hub 请求 "hello-world"镜像。
  + docker daemon 从这个镜像，创建一个新的容器并产生了你看到的输出内容。
  + docker daemon 流输出到 docker 客户端 ，并输出到你的终端。

下面我们通过一张图，来看一下 docker 运行的流程:



从这张图我们可以清楚了解到 docker 的运转流程。

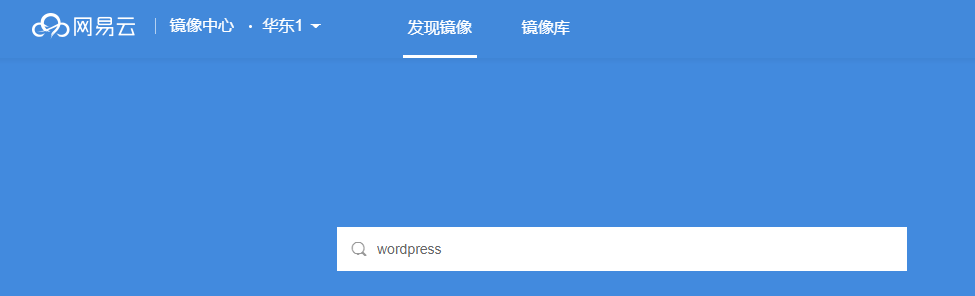
Docker 安装 WordPress

好了上面介绍了那么多概念，其实都是为这边工作做铺垫。我们不能知其然，而不知其所以然！ 我们可以在网上搜索相应的教程.

下面我们来说说使用 Docker 方式部署：

**0x01 寻找 docker 镜像**

首先我们打开[网易镜像中心](https://c.163.com/hub#/m/home/) ( 直接访问的话，需要先登录), 根据目标进行检索。我们知道常规的 wordpress 是基于 php 环境的，那么我们需要 lmap 的环境。这边我们直接去搜索 wordpress 好了，如果 wordpress 存在则说明该镜像已经集成了基本的运行环境，我们就不需要额外再去寻找了。

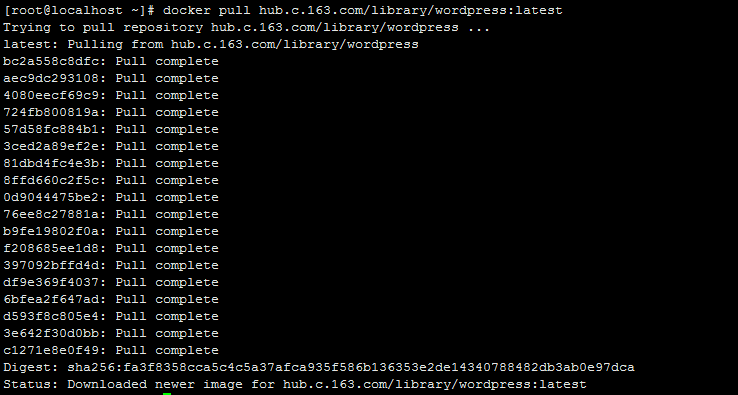




**0x02 将 docker 镜像拉倒本地**

幸运的是，我们找到了 wordpress 的镜像，我们直接复制右上角的链接：  
docker pull hub.c.163.com/library/wordpress:latest

拷贝到 linux 环境进行运行。运行效果如下：

好了，下面我们输入 docker images查看目前已经安装的镜像文件ok ，没问题。

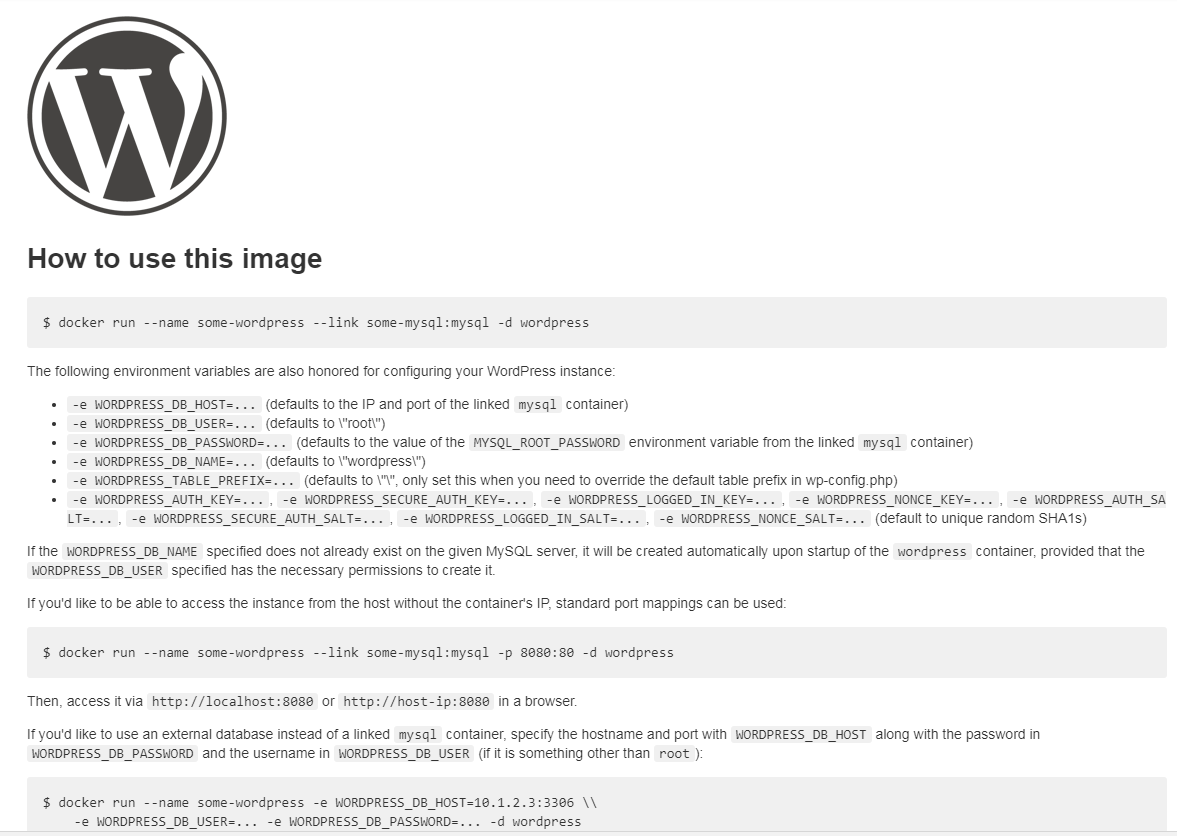
enter image description here

那下面我们该如何运行这个镜像文件呢？

**0x03 阅读 docker 镜像说明**

聪明的人都知道在 linux 上面不懂得都请求 man 男人，这边我们不懂得 当然去请求我们得官网啦。这边就特指了我们刚才下载镜像的网页。我们去看看 wordpress 说明：

很容易，我们看到 wordpress，

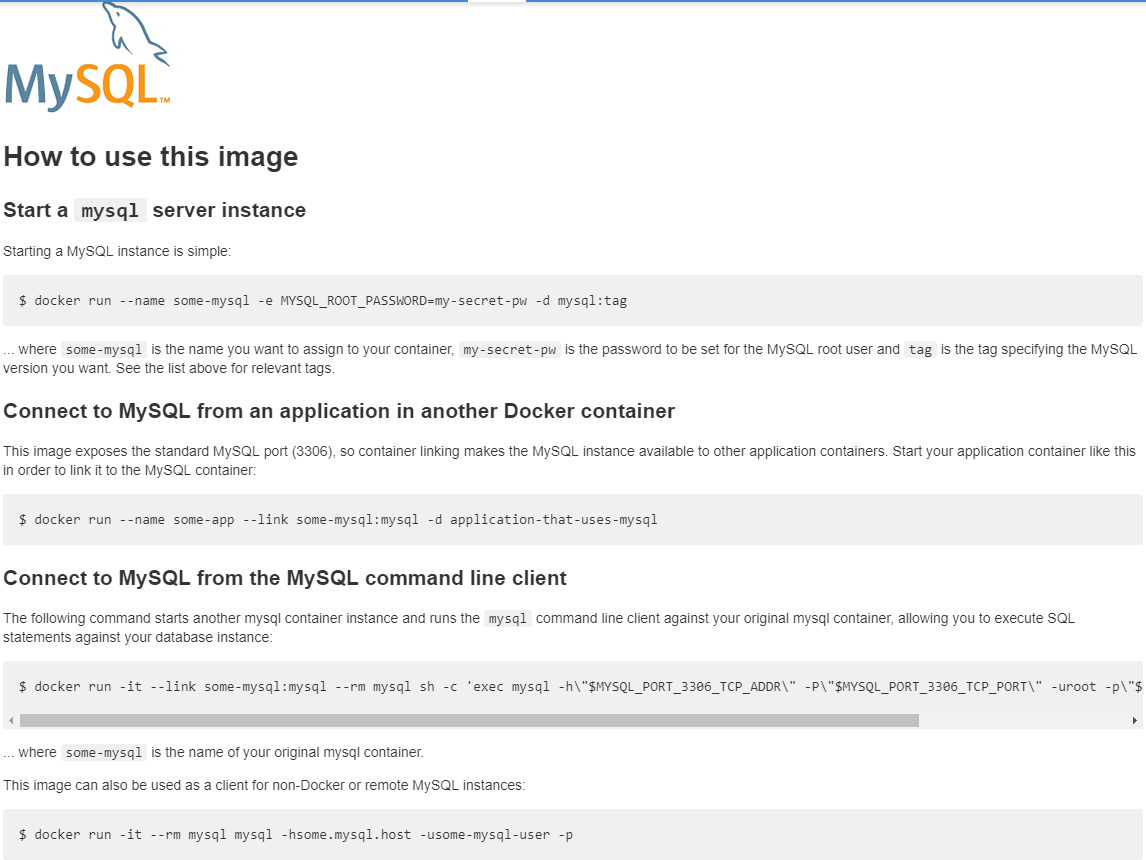


我们来分析一下 WordPress 他的参数说明：

* WORDPRESS\_DB\_HOST：wordpress 连接数据库的参数默认 ip：port
* WORDPRESS\_DB\_USER：wordpress 连接数据库的用户
* WORDPRESS\_DB\_PASSWORD：wordpress 连接数据库的密码
* WORDPRESS\_DB\_NAME：wordpress 数据库的名称默认为：wordpress

我们发现还缺失 mysql, 一样的方法查找 mysql 镜像，并拉取到本地。





我们同样来分析一下 Mysql 的参数说明：

* MYSQL\_ROOT\_PASSWORD：设置 mysql 数据库的密码。
* MYSQL\_DATABASE：设置 mysql 的数据库名称。
* MYSQL\_USER：设置 mysql 的数据库用户。

**0x04 运行 docker 镜像**

下面我们就输入 我们在网站说明找到的 wordpress 说明指令

**启动 Mysql 指令：**

配置 mysql 的密码，以及 mysql 数据库名称，并且 -p 设置端口映射，将本机端口和容器内部端口对应起来：

docker run -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 -e MYSQL\_DATABASE=wordpress -d hub.c.163.com/library/mysql

enter image description here

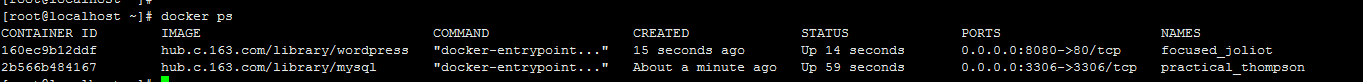
**启动 wordpress 指令：**

配置启动的 wordpress 对应数据库地址以及端口，数据库用户名、密码以及端口映射，将本地 8080 端口映射到容器内部的 80 端口：

docker run -e WORDPRESS\_DB\_HOST=172.16.8.95:3306 -e WORDPRESS\_DB\_USER=root -e WORDPRESS\_DB\_PASSWORD=123456 -p 8080:80 -d hub.c.163.com/library/wordpress

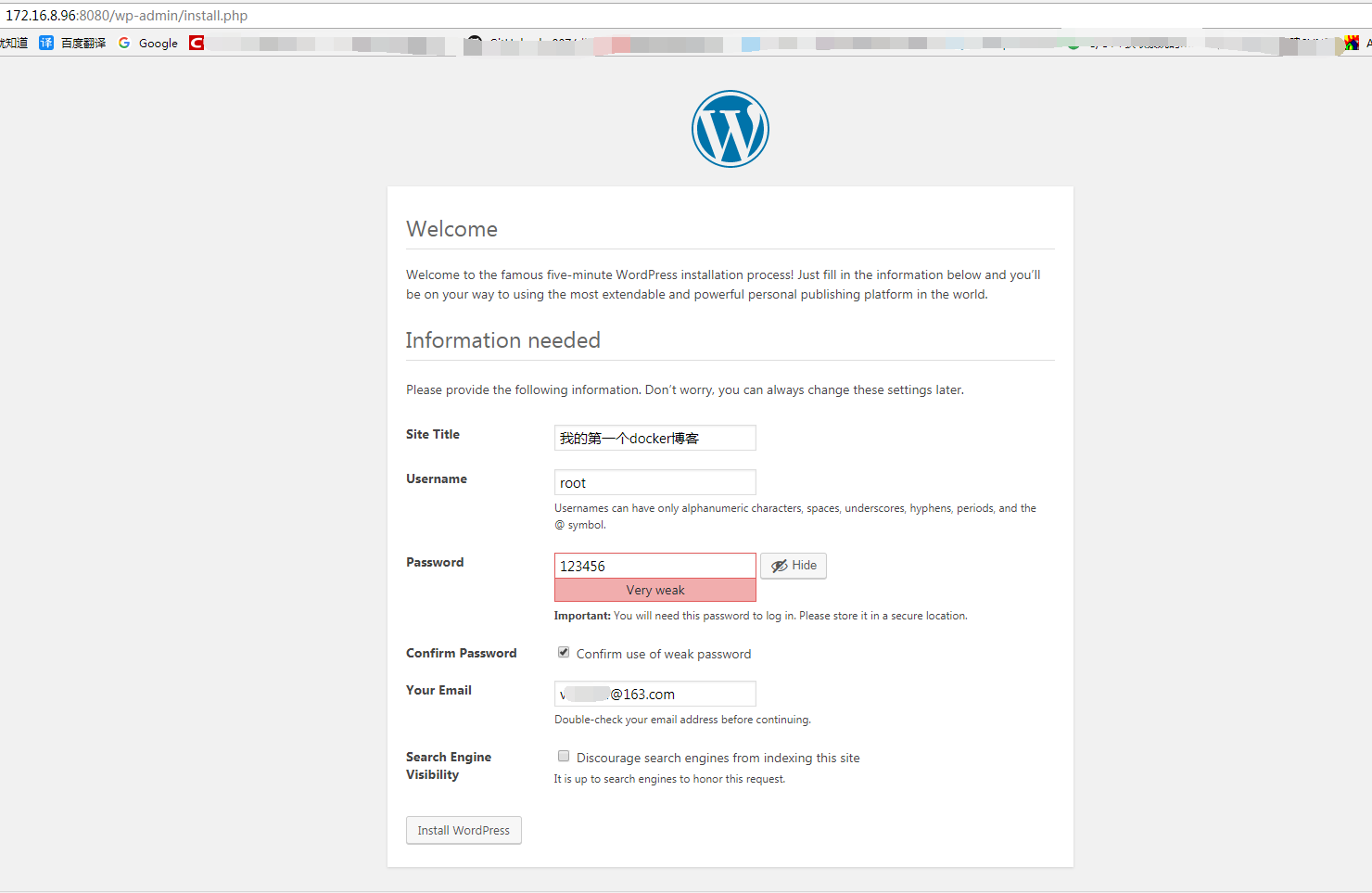
enter image description here

**查看进程情况：**

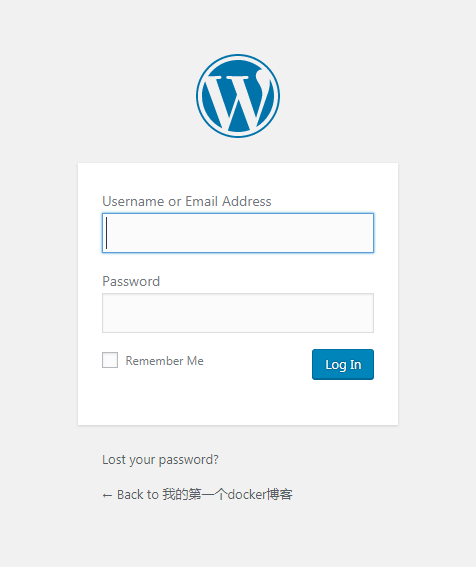


**0x05 访问 WordPress 主页**

至此，我们前面做的工作已经完成，下面我们开始访问网页，注意把后台服务器防火墙关闭，关闭指令：service firewalld stop；然后，输入博客搭建环境的地址: 172.16.8.96:8080,可以看到他自动跳转到 初始安装界面如下， 用户根据自己需求， 设置博客名称，博客管理员账户、密码。



然后点击 install wordpress，输入刚才的用户名，密码



点击登录，ok，咱们第一个基于 docker 的博客搭建完毕。下面 你可以在仪表盘设置自己想要的风格和插件咯。



咦，你的 wordpress 已经可以跑起来了吗？ 是的 就是这么简单 ，是的 就是这么方便，五个步骤就可以完成。下次在部署和迁移的时候是不是更方便呢？我们是时候学会更简单的方法部署我们需要的环境了！

结束语

通过本篇 docker 的学习，相信作为从未接触过的人而言 会有一种醍醐灌顶、豁然开朗的效果。如果将你的项目发布部署全部通过docker，那么对于运维人员而言是多么幸福的事情，与人方便，与已方便！就像这篇关于 wordpress 部署的方式和传统的 LAMP + WordPress 的部署方式，哪种更轻便，相信聪明的你，一定已经感受到了 docker 的魅力。

如果觉得本篇文章对您有用，欢迎点赞+评论。您的支持，是我源源不断的动力。最后还是以 docker 最为经典一句话结束本篇 Chat：

**Build Once, Works Everywhere！！！**

**附录：** **Docker常用指令汇总**

docker images：查看已经安装的 docker 镜像。  
docker image rm image\_id：删除镜像。  
docker pull：获取镜像。  
docker search：从仓库查找镜像文件。  
docker exec –it fID bash：进入 docker 容器内部。

docker run：运行 docker 镜像。  
docker ps：查看运行容器进程。  
docker run –d：在后台运行 docker 镜像。  
docker run –p：3306:3306 端口映射。  
docker stop ID：停止正在运行的容器。

更多指令查看：  
http://www.runoob.com/docker/docker-command-manual.html