

## Docker 教程



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。

### 谁适合阅读本教程？

本教程适合运维工程师及后端开发人员，通过本教程你可以一步一步了解 Docker 的使用。

### 阅读本教程前，您需要了解的知识

在阅读本教程前，你需要掌握 Linux 的常用命令。你可以通过本站的 [Linux 教程](#) 来学习相关命令。

### Docker 的应用场景

Web 应用的自动化打包和发布。

自动化测试和持续集成、发布。

在服务型环境中部署和调整数据库或其他的后台应用。

从头编译或者扩展现有的 OpenShift 或 Cloud Foundry 平台来搭建自己的 PaaS 环境。

### 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。

### 相关链接

Docker 官网：<http://www.docker.com>

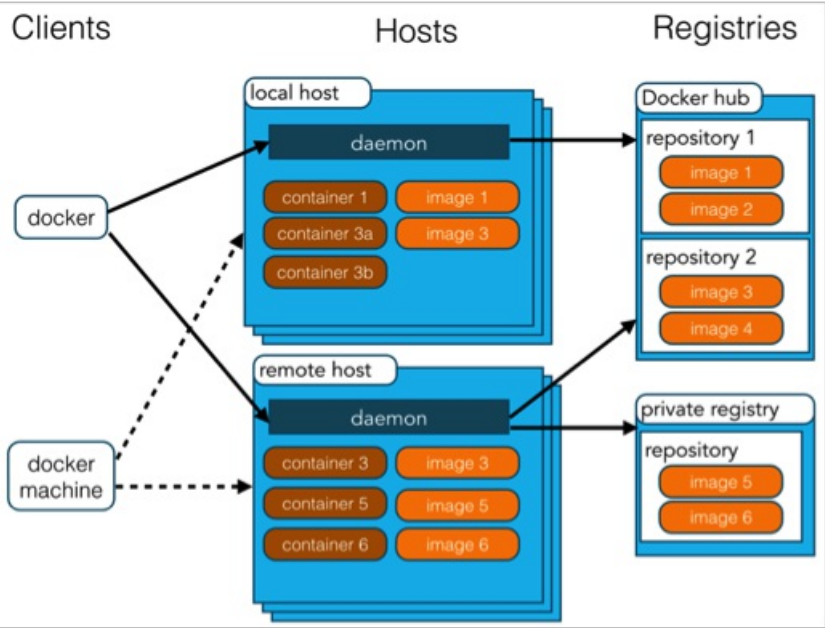
Github Docker 源码：<https://github.com/docker/docker>



Docker 架构

Docker 使用客户端-服务器 (C/S) 架构模式，使用远程API来管理和创建Docker容器。  
Docker 容器通过 Docker 镜像来创建。  
容器与镜像的关系类似于面向对象编程中的对象与类。

Docker	面向对象
容器	对象
镜像	类



Docker 镜像(Images)	Docker 镜像是用于创建 Docker 容器的模板。
Docker 容器(Container)	容器是独立运行的一个或一组应用。
Docker 客户端(Client)	Docker 客户端通过命令行或者其他工具使用 Docker API ( <a href="https://docs.docker.com/reference/api/docker_remote_api">https://docs.docker.com/reference/api/docker_remote_api</a> ) 与 Docker 的守护进程通信。
Docker 主机(Host)	一个物理或者虚拟的机器用于执行 Docker 守护进程和容器。

Docker 仓库(Registry)	Docker 仓库用来保存镜像，可以理解为代码控制中的代码仓库。 Docker Hub( <a href="https://hub.docker.com">https://hub.docker.com</a> ) 提供了庞大的镜像集合供使用。
Docker Machine	Docker Machine是一个简化Docker安装的命令行工具，通过一个简单的命令行即可在相应的平台上安装Docker，比如VirtualBox、Digital Ocean、Microsoft Azure。

[Docker port 命令](#)

[Docker images 命令](#)

[点我分享笔记](#)

[反馈/建议](#)

Copyright © 2013-2018 菜鸟教程 [runoob.com](http://runoob.com) All Rights Reserved. 备案号：闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[Docker Hello World](#)

[Docker 容器使用](#)

## Ubuntu Docker 安装

Docker 支持以下的 Ubuntu 版本：

Ubuntu Precise 12.04 (LTS)

Ubuntu Trusty 14.04 (LTS)

Ubuntu Wily 15.10

其他更新的版本.....

### 前提条件

Docker 要求 Ubuntu 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 Ubuntu 版本是否支持 Docker。

通过 `uname -r` 命令查看你当前的内核版本

```
runoob@runoob:~$ uname -r
```

```
runoob@runoob:~$ uname -r
4.2.0-16-generic
runoob@runoob:~$
```

### 使用脚本安装 Docker

#### 1、获取最新版本的 Docker 安装包

```
runoob@runoob:~$ wget -qO- https://get.docker.com/ | sh
```

输入当前用户的密码后，就会下载脚本并且安装Docker及依赖包。

```
[sudo] password for runoob:
apparmor is enabled in the kernel, but apparmor_parser missing
+ sudo -E sh -c sleep 3; apt-get update
Hit http://hk.archive.ubuntu.com wily InRelease
Get:1 http://hk.archive.ubuntu.com wily-updates InRelease [65.9 kB]
Get:2 http://security.ubuntu.com wily-security InRelease [65.9 kB]
Get:3 http://hk.archive.ubuntu.com wily-backports InRelease [65.9 kB]
Get:4 http://hk.archive.ubuntu.com wily/main Sources [1,118 kB]
Get:5 http://security.ubuntu.com wily-security/restricted Sources [2,854 B]
Get:6 http://security.ubuntu.com wily-security/universe Sources [11.2 kB]
Get:7 http://hk.archive.ubuntu.com wily/restricted Sources [7,181 B]
Get:8 http://hk.archive.ubuntu.com wily/universe Sources [7,238 kB]
Get:9 http://security.ubuntu.com wily-security/main Sources [44.2 kB]
Get:10 http://security.ubuntu.com wily-security/main amd64 Packages [143 kB]
Get:11 http://security.ubuntu.com wily-security/multiverse Sources [2,782 B]
Get:12 http://security.ubuntu.com wily-security/restricted amd64 Packages [10.9 kB]
Get:13 http://security.ubuntu.com wily-security/multiverse amd64 Packages [6,253 B]
Get:14 http://security.ubuntu.com wily-security/universe amd64 Packages [50.2 kB]
Get:15 http://security.ubuntu.com wily-security/main i386 Packages [139 kB]
Get:16 http://security.ubuntu.com wily-security/restricted i386 Packages [10.8 kB]
Get:17 http://security.ubuntu.com wily-security/universe i386 Packages [50.2 kB]
Get:18 http://security.ubuntu.com wily-security/main Translation-en [69.2 kB]
Get:19 http://security.ubuntu.com wily-security/multiverse Translation-en [2,806 B]
Get:20 http://security.ubuntu.com wily-security/multiverse i386 Packages [6,430 B]
Get:21 http://security.ubuntu.com wily-security/restricted Translation-en [2,666 B]
Get:22 http://security.ubuntu.com wily-security/universe Translation-en [33.1 kB]
```

安装完成后有个提示：

If you would like to use Docker as a non-root user, you should now consider

adding your user to the "docker" group with something like:

```
sudo usermod -aG docker runoob
```

Remember that you will have to log out and back in for this to take effect!

当要以非root用户可以直接运行docker时，需要执行 **sudo usermod -aG docker runoob** 命令，然后重新登陆，否则会有如下报错

## 2、启动docker 后台服务

```
runoob@runoob:~$ sudo service docker start
```

```
runoob@runoob:~$ sudo service docker start
runoob@runoob:~$ ps -ef|grep docker
root      11123      1      15:33 ?        00:00:00 /usr/bin/docker daemon -H fd://
root      11126 11123      0 15:33 ?        00:00:00 docker-containerd -l /var/run/docker/libcontainerd/docker-containerd.sock --runtime docker-runc
runoob    11180 11743      0 15:33 pts/1    00:00:00 grep --color=auto docker
```

## 3、测试运行hello-world

```
runoob@runoob:~$ docker run hello-world
```

## 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） 或者 %programdata%\docker\config\daemon.json（Windows） 来配置 Daemon。

请在该配置文件中加入（没有该文件的话，请先建一个）：

```
{
  "registry-mirrors": ["http://hub-mirror.c.163.com"]
}
```

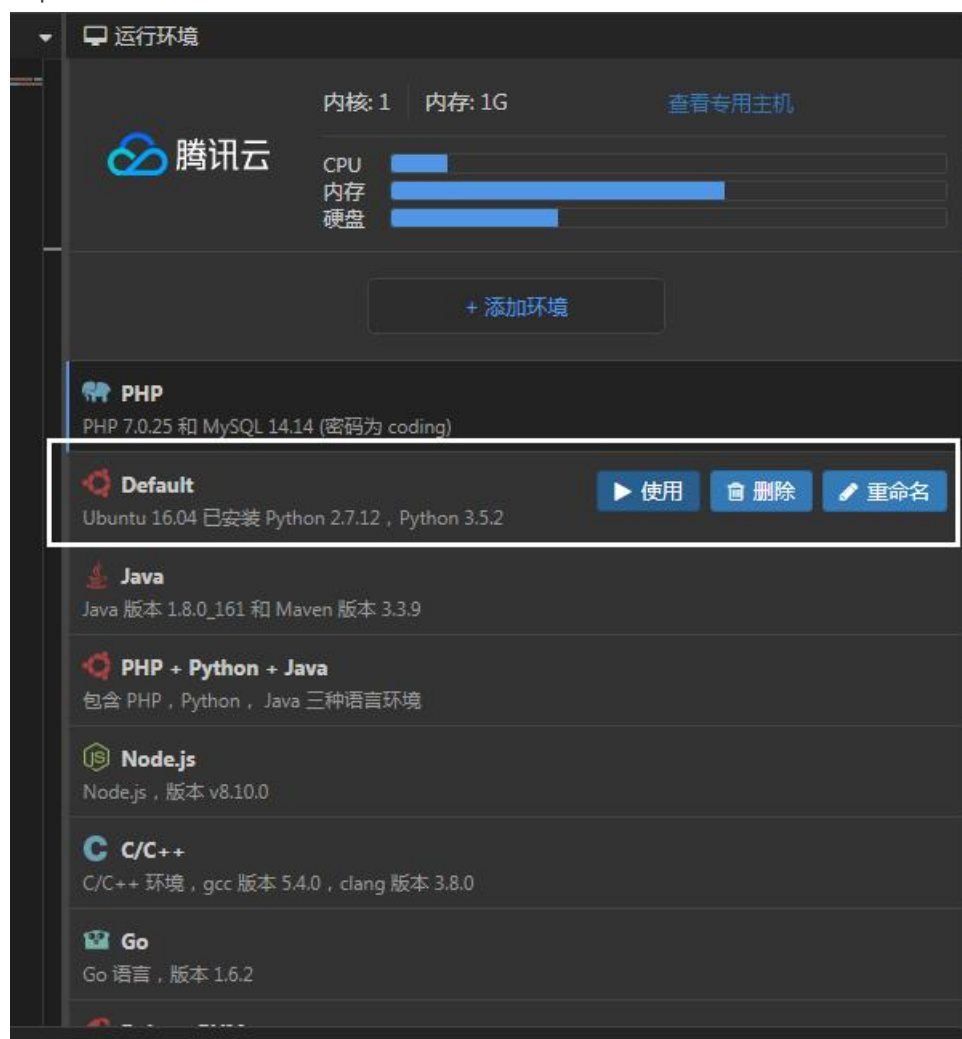
}

## 在 Cloud Studio 中运行 Docker

下面我们介绍如何在 Cloud Studio 中安装、使用 Docker:

step1: 访问 [Cloud Studio](#), 注册/登录账户。

step2: 在右侧的运行环境菜单选择: "ubuntu"



step3: 在下方的终端执行命令, 获取最新版本的 Docker 安装包并执行安装:

```
wget -qO- https://get.docker.com/ | sh
```

step4: 启动 docker 后台服务:

```
sudo service docker start
```

step5: 测试运行 hello-world:

```
sudo docker run hello-world
```

☐ Docker Hello World

Docker 容器使用 ☐



2 篇笔记  
#2

☐ 写笔记



Ubuntu 16.04 安装 Docker

1.选择国内的云服务商，这里选择阿里云为例

```
curl -sSL http://acs-public-mirror.oss-cn-hangzhou.aliyuncs.com/docker-engine/internet | sh -
```

2.安装所需要的包

```
sudo apt-get install linux-image-extra-$(uname -r) linux-image-extra-virtual
```

3.添加使用 HTTPS 传输的软件包以及 CA 证书

```
sudo apt-get update

sudo apt-get install apt-transport-https ca-certificates
```

4.添加GPG密钥

```
sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
```

5.添加软件源

```
echo "deb https://apt.dockerproject.org/repo ubuntu-xenial main" | sudo tee /etc/apt/sources.list.d/docker.list
```

6.添加成功后更新软件包缓存

```
sudo apt-get update
```

7.安装docker

```
sudo apt-get install docker-engine
```

8.启动 docker

```
sudo systemctl enable docker

sudo systemctl start docker
```

欧迪芬3个月前 [06-23]

#1



Ubuntu 18.04 安装 Docker-ce

1.更换国内软件源，推荐中国科技大学的源，稳定速度快（可选）

```
sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak

sudo sed -i 's/archive.ubuntu.com/mirrors.ustc.edu.cn' /etc/apt/sources.list

sudo apt update
```

2.安装需要的包

```
sudo apt install apt-transport-https ca-certificates software-properties-common curl
```

3.添加 GPG 密钥，并添加 Docker-ce 软件源，这里还是以中国科技大学的 Docker-ce 源为例

```
curl -fsSL https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu/gpg | sudo apt-key add -
```

```
sudo add-apt-repository "deb [arch=amd64] https://mirrors.ustc.edu.cn/docker-ce/linux/ubuntu \
$(lsb_release -cs) stable"
```

4.添加成功后更新软件包缓存

```
sudo apt update
```

5.安装 Docker-ce

```
sudo apt install docker-ce
```

6.设置开机自启动并启动 Docker-ce（安装成功后默认已设置并启动，可忽略）

```
sudo systemctl enable docker

sudo systemctl start docker
```

7.测试运行

```
sudo docker run hello-world
```

8.添加当前用户到 docker 用户组，可以不用 sudo 运行 docker（可选）

```
sudo groupadd docker

sudo usermod -aG docker $USER
```

9.测试添加用户组（可选）

```
docker run hello-world
```

路人码农3个月前 (07-16)

反馈/建议



# CentOS Docker 安装

Docker支持以下的CentOS版本：

- CentOS 7 (64-bit)
- CentOS 6.5 (64-bit) 或更高的版本

## 前提条件

目前，CentOS 仅发行版本中的内核支持 Docker。

Docker 运行在 CentOS 7 上，要求系统为64位、系统内核版本为 3.10 以上。

Docker 运行在 CentOS-6.5 或更高的版本的 CentOS 上，要求系统为64位、系统内核版本为 2.6.32-431 或者更高版本。

## 使用 yum 安装（CentOS 7下）

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 Docker。

通过 **uname -r** 命令查看你当前的内核版本

```
[root@runoob ~]# uname -r 3.10.0-327.el7.x86_64
```

## 安装 Docker

从 2017 年 3 月开始 docker 在原来的基础上分为两个分支版本: Docker CE 和 Docker EE。

Docker CE 即社区免费版，Docker EE 即企业版，强调安全，但需付费使用。

本文介绍 Docker CE 的安装使用。

移除旧的版本：

```
$ sudo yum remove docker \
                          docker-client \
                          docker-client-latest \
                          docker-common \
                          docker-latest \
                          docker-latest-logrotate \
                          docker-logrotate \
                          docker-selinux \
                          docker-engine-selinux \
                          docker-engine
```

安装一些必要的系统工具：

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

添加软件源信息：

```
sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

更新 yum 缓存：

```
sudo yum makecache fast
```

安装 Docker-ce:

```
sudo yum -y install docker-ce
```

启动 Docker 后台服务



```
sudo systemctl start docker
```

测试运行 **hello-world**

```
[root@runoob ~]# docker run hello-world
```

□

由于本地没有**hello-world**这个镜像，所以会下载一个**hello-world**的镜像，并在容器内运行。

## 使用脚本安装 **Docker**

- 1、使用 `sudo` 或 `root` 权限登录 **Centos**。
- 2、确保 `yum` 包更新到最新。

```
$ sudo yum update
```

- 3、执行 **Docker** 安装脚本。

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

执行这个脚本会添加 `docker.repo` 源并安装 **Docker**。

- 4、启动 **Docker** 进程。

```
sudo systemctl start docker
```

- 5、验证 `docker` 是否安装成功并在容器中执行一个测试的镜像。

```
$ sudo docker run hello-world  
  
docker ps
```

到此，**Docker** 在 **CentOS** 系统的安装完成。

## 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）或者 `%programdata%\docker\config\daemon.json`（Windows）来配置 **Daemon**。

请在该配置文件中加入（没有该文件的话，请先建一个）：

```
{  
  
  "registry-mirrors": ["http://hub-mirror.c.163.com"]  
}
```

```
}
```

## 删除 Docker CE

执行以下命令来删除 Docker CE:

```
$ sudo yum remove docker-ce

$ sudo rm -rf /var/lib/docker
```

[☐ Docker 教程](#)

[Windows Docker 安装](#) ☐

[☐ 点我分享笔记](#)

反馈/建议

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号: 闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[☐ CentOS Docker 安装](#)

[Docker Hello World](#) ☐

## Windows Docker 安装

### win7、win8 系统

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

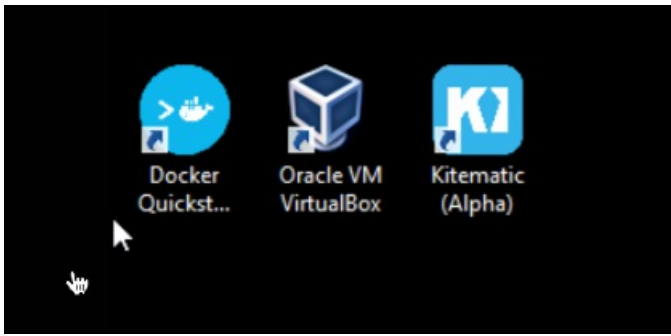
Docker Compose. 用来运行docker-compose命令

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

下载完成之后直接点击安装, 安装成功后, 桌边会出现三个图标, 入下图所示:



点击 Docker QuickStart 图标来启动 Docker Toolbox 终端。

如果系统显示 User Account Control 窗口来运行 VirtualBox 修改你的电脑，选择 Yes。

```
MINGW32/c/Users/M
export DOCKER_HOST=tcp://192.168.59.103:2376
export DOCKER_CERT_PATH='C:\Users\M\.boot2docker\certs\boot2docker-vm\'
export DOCKER_TLS_VERIFY=1

IP address of docker VM:
192.168.59.103

setting environment variables ...
Writing C:\Users\M\.boot2docker\certs\boot2docker-vm\ca.pem
Writing C:\Users\M\.boot2docker\certs\boot2docker-vm\cert.pem
Writing C:\Users\M\.boot2docker\certs\boot2docker-vm\key.pem
export DOCKER_HOST=tcp://192.168.59.103:2376
export DOCKER_CERT_PATH='C:\Users\M\.boot2docker\certs\boot2docker-vm\'
export DOCKER_TLS_VERIFY=1

You can now use 'docker' directly, or 'boot2docker ssh' to log into the VM.
Welcome to Git (version 1.9.5-preview20150319)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

MERNEALE ~
$
```

\$ 符号那你可以输入以下命令来执行。

```
$ docker run hello-world

Unable to find image 'hello-world:latest' locally

Pulling repository hello-world
91c95931e552: Download complete
a8219747be10: Download complete

Status: Downloaded newer image for hello-world:latest

Hello from Docker.

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker Engine CLI client contacted the Docker Engine daemon.
2. The Docker Engine daemon pulled the "hello-world" image from the Docker Hub.

   (Assuming it was not already locally available.)
3. The Docker Engine daemon created a new container from that image which runs the

   executable that produces the output you are currently reading.
4. The Docker Engine daemon streamed that output to the Docker Engine CLI client, which sent it
```

to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

For more examples and ideas, visit:

<https://docs.docker.com/userguide/>

## Win10 系统

现在 Docker 有专门的 Win10 专业版系统的安装包，需要开启Hyper-V。

### 开启 Hyper-V

- 程序和功能
- 启用或关闭Windows功能
- 选中Hyper-V
- 

### 1、安装 Toolbox

最新版 Toolbox 下载地址：<https://www.docker.com/get-docker>

点击 [Get Docker Community Edition](#)，并下载 Windows 的版本：

- 
- 

### 2、运行安装文件

双击下载的 Docker for Windows Installer 安装文件，一路 Next，点击 Finish 完成安装。



- 安装完成后，Docker 会自动启动。通知栏上会出现个小鲸鱼的图标，这表示 Docker 正在运行。桌边也会出现三个图标，入下图所示：
- 我们可以在命令行执行 `docker version` 来查看版本号，`docker run hello-world` 来载入测试镜像测试。
- 如果没启动，你可以在 Windows 搜索 Docker 来启动：

- 启动后，也可以在通知栏上看到小鲸鱼图标：
- 

## 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）或者 `%programdata%\docker\config\daemon.json`（Windows）来配置 Daemon。

请在该配置文件中加入（没有该文件的话，请先建一个）：

```
{  
  
  "registry-mirrors": ["http://hub-mirror.c.163.com"]  
  
}
```

[☐ 点我分享笔记](#)

反馈/建议



[☐ Docker 资源汇总](#)

# MacOS Docker 安装

## 使用 Homebrew 安装

macOS 我们可以使用 Homebrew 来安装 Docker。  
Homebrew 的 Cask 已经支持 Docker for Mac，因此可以很方便的使用 Homebrew Cask 来进行安装：

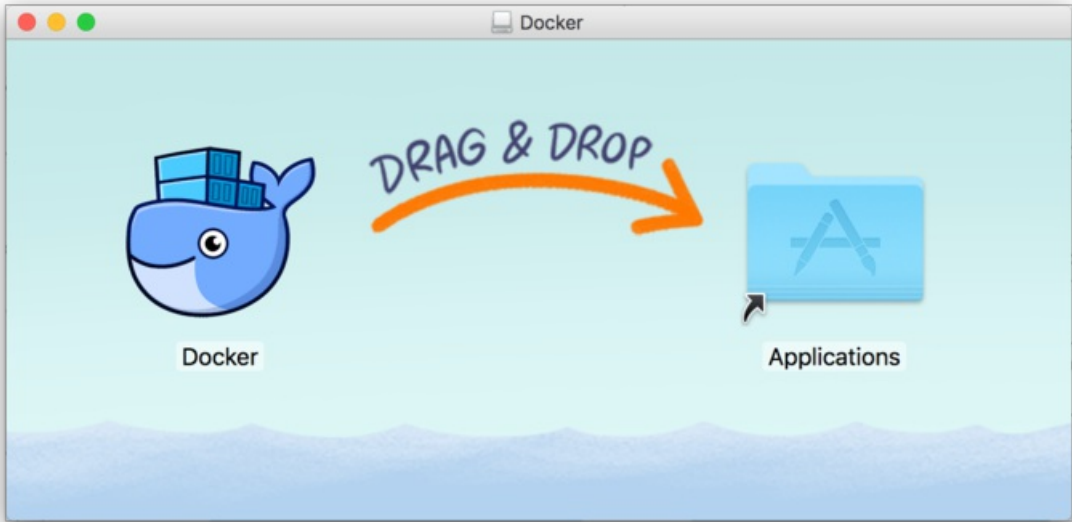
```
$ brew cask install docker  
  
==> Creating Caskroom at /usr/local/Caskroom  
  
==> We'll set permissions properly so we won't need sudo in the future  
  
Password:          # 输入 macOS 密码  
  
==> Satisfying dependencies  
  
==> Downloading https://download.docker.com/mac/stable/21090/Docker.dmg  
  
##### 100.0%  
  
==> Verifying checksum for Cask docker  
  
==> Installing Cask docker  
  
==> Moving App 'Docker.app' to '/Applications/Docker.app'.  
  
&#x1f37a;  docker was successfully installed!
```

在载入 Docker app 后，点击 Next，可能会询问你的 macOS 登陆密码，你输入即可。之后会弹出一个 Docker 运行的提示窗口，状态栏上也有有个小鲸鱼的图标（🐳）。

## 手动下载安装

如果需要手动下载，请点击以下链接下载 [Stable](#) 或 [Edge](#) 版本的 Docker for Mac。

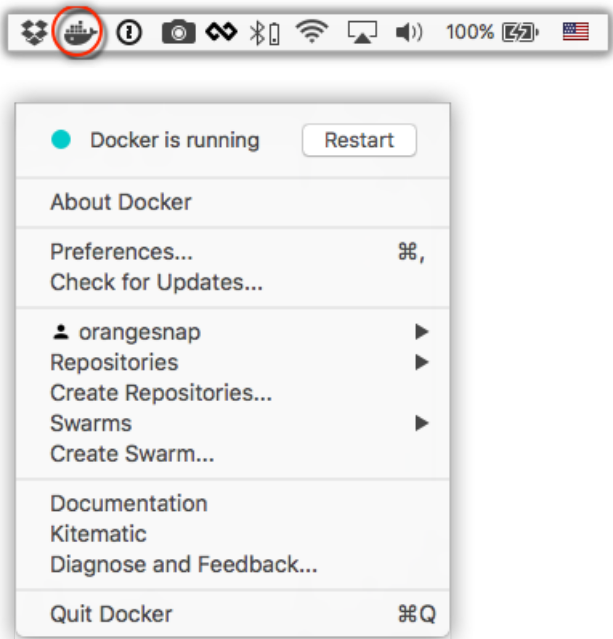
如同 macOS 其它软件一样，安装也非常简单，双击下载的 .dmg 文件，然后将鲸鱼图标拖拽到 Application 文件夹即可。



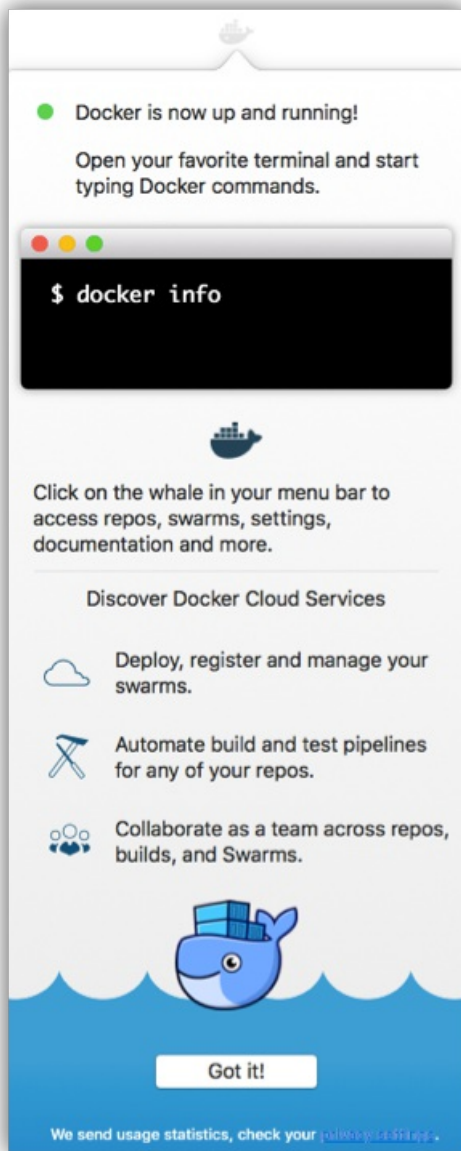
从应用中找到 Docker 图标并点击运行。可能会询问 macOS 的登陆密码，输入即可。



点击顶部状态栏中的鲸鱼图标会弹出操作菜单。



第一次点击图标，可能会看到这个安装成功的界面，点击 "Got it!" 可以关闭这个窗口。



启动终端后，通过命令可以检查安装后的 Docker 版本。

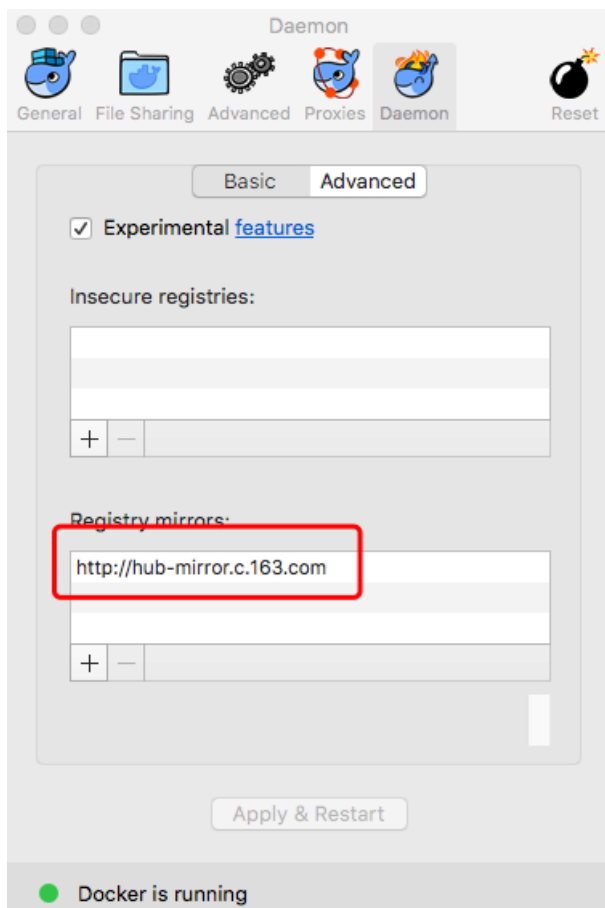
```
$ docker --version
```

```
Docker version 17.09.1-ce, build 19e2cf6
```

## 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>。

在任务栏点击 Docker for mac 应用图标 -> Preferences... -> Daemon -> Registry mirrors。在列表中填写加速器地址即可。修改完成之后，点击 Apply & Restart 按钮，Docker 就会重启并应用配置的镜像地址了。



之后我们可以通过 `docker info` 来查看是否配置成功。

```
$ docker info

...

Registry Mirrors:

  http://hub-mirror.c.163.com

Live Restore Enabled: false
```

[□ Docker 资源汇总](#)

[□ 点我分享笔记](#)

[反馈/建议](#)

Copyright © 2013-2018 菜鸟教程 [runoob.com](http://runoob.com) All Rights Reserved. 备案号: 闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[□ Windows Docker 安装](#)

[Ubuntu Docker 安装](#) [□](#)



# Docker Hello World

Docker 允许你在容器内运行应用程序，使用 **docker run** 命令来在容器内运行一个应用程序。

输出Hello world

```
runoob@runoob:~$ docker run ubuntu:15.10 /bin/echo "Hello world"
```

```
Hello world
```

```
runoob@runoob:~$ docker run ubuntu:15.10 /bin/echo "Hello world"
Hello world
```

各个参数解析：

**docker:** Docker 的二进制执行文件。

**run:**与前面的 **docker** 组合来运行一个容器。

**ubuntu:15.10**指定要运行的镜像，Docker首先从本地主机上查找镜像是否存在，如果不存在，Docker 就会从镜像仓库 Docker Hub 下载公共镜像。

**/bin/echo "Hello world":** 在启动的容器里执行的命令

以上命令完整的意思可以解释为：Docker 以 **ubuntu15.10** 镜像创建一个新容器，然后在容器里执行 **bin/echo "Hello world"**，然后输出结果。

## 运行交互式的容器

我们通过docker的两个参数 **-i -t**，让docker运行的容器实现"对话"的能力

```
runoob@runoob:~$ docker run -i -t ubuntu:15.10 /bin/bash
```

```
root@dc0050c79503:/#
```

各个参数解析：

**-t:**在新容器内指定一个伪终端或终端。

**-i:**允许你对容器内的标准输入 (STDIN) 进行交互。

此时我们已进入一个 **ubuntu15.10**系统的容器

我们尝试在容器中运行命令 **cat /proc/version**和**ls**分别查看当前系统的版本信息和当前目录下的文件列表

```
root@dc0050c79503:/# cat /proc/version
Linux version 4.2.0-35-generic (buildd@lcy01-07) (gcc version 5.2.1 20151003 (Ubuntu 5.2.1-21ubuntu2) ) #19-Ubuntu SMP Thu
Oct 8 15:35:06 UTC 2015
root@dc0050c79503:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys usr var
```

我们可以通过运行**exit**命令或者使用**CTRL+D**来退出容器。

## 启动容器（后台模式）

使用以下命令创建一个以进程方式运行的容器

```
runoob@runoob:~$ docker run -d ubuntu:15.10 /bin/sh -c "while true; do echo hello world; sleep 1; done"
```

```
2b1b7a428627c51ab8810d541d759f072b4fc75487eed05812646b8534a2fe63
```

```
runoob@runoob:~$ docker run -d ubuntu:15.10 /bin/sh -c "while true; do echo hello world; sleep 1; done"
2b1b7a428627c51ab8810d541d759f072b4fc75487eed05812646b8534a2fe63
```

在输出中，我们没有看到期望的"hello world"，而是一串长字符

2b1b7a428627c51ab8810d541d759f072b4fc75487eed05812646b8534a2fe63

这个长字符串叫做容器ID，对每个容器来说都是唯一的，我们可以通过容器ID来查看对应的容器发生了什么。

首先，我们需要确认容器有在运行，可以通过 **docker ps** 来查看

```
runoob@runoob:~$ docker ps
```

```
runoob@runoob:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
2b1b7a428627   ubuntu:15.10  "/bin/sh -c 'while tr"  6 minutes ago  Up 6 minutes  0.0.0.0:80->0.0.0.0:80   amazing_cor1
```

**CONTAINER ID:**容器ID

**NAMES:**自动分配的容器名称

在容器内使用**docker logs**命令，查看容器内的标准输出

```
runoob@runoob:~$ docker logs 2b1b7a428627
```

```
runoob@runoob:~$ docker logs 2b1b7a428627
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
```

```
runoob@runoob:~$ docker logs amazing_cor1
```

```
runoob@runoob:~$ docker logs amazing_cor1
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
```

## 停止容器

我们使用 **docker stop** 命令来停止容器：

```
runoob@runoob:~$ docker stop 2b1b7a428627
2b1b7a428627
```

通过**docker ps**查看，容器已经停止工作：

```
runoob@runoob:~$ docker ps
```

```
runoob@runoob:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
runoob@runoob:~$
```

也可以用下面的命令来停止：

```
runoob@runoob:~$ docker stop amazing_cori
```

[☐ Windows Docker 安装](#)[Ubuntu Docker 安装 ☐](#)[☐ 点我分享笔记](#)[反馈/建议](#)

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1

[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)[☐ Ubuntu Docker 安装](#)[Docker 镜像使用 ☐](#)

## Docker 容器使用

### Docker 客户端

docker 客户端非常简单 ,我们可以直接输入 **docker** 命令来查看到 Docker 客户端的所有命令选项。

```
runoob@runoob:~# docker
```

☐

可以通过命令 **docker command --help** 更深入的了解指定的 Docker 命令使用方法。

例如我们要查看 **docker stats** 指令的具体使用方法：

```
runoob@runoob:~# docker stats --help
```

```
runoob@runoob:/root$ docker stats --help
Usage:  docker stats [OPTIONS] [CONTAINER...]

Display a live stream of container(s) resource usage statistics

  -a, --all           show all containers (default shows just running)
  --help             Print usage
  --no-stream        Disable streaming stats and only pull the first result
```

### 运行一个web应用

前面我们运行的容器并没有一些什么特别的用处。

接下来让我们尝试使用 **docker** 构建一个 **web** 应用程序。

我们将在**docker**容器中运行一个 **Python Flask** 应用来运行一个web应用。

```
runoob@runoob:~# docker pull training/webapp # 载入镜像
```

```
runoob@runoob:~# docker run -d -P training/webapp python app.py
```

```
runoob@runoob:~$ docker run -d -P training/webapp python app.py
4ecb9ce5113ded09117a94462fee760c89f9db7bfc10365ca20f2d5a4430871b
runoob@runoob:~$
```

参数说明：

**-d**:让容器在后台运行。

**-P**:将容器内部使用的网络端口映射到我们使用的主机上。

## 查看 WEB 应用容器

使用 **docker ps** 来查看我们正在运行的容器：

```
runoob@runoob:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	...	PORTS
d3d5e39ed9d3	training/webapp	"python app.py"	...	0.0.0.0:32768->5000/tcp

这里多了端口信息。

```
PORTS

0.0.0.0:32769->5000/tcp
```

Docker 开放了 5000 端口（默认 Python Flask 端口）映射到主机端口 32769 上。

这时我们可以通过浏览器访问WEB应用

□

我们也可以通过 **-p** 参数来设置不一样的端口：

```
runoob@runoob:~$ docker run -d -p 5000:5000 training/webapp python app.py
```

**docker ps**查看正在运行的容器

```
runoob@runoob:~# docker ps
```

CONTAINER ID	IMAGE	PORTS	NAMES
bf08b7f2cd89	training/webapp	... 0.0.0.0:5000->5000/tcp	wizardly_chandrasekhar
d3d5e39ed9d3	training/webapp	... 0.0.0.0:32768->5000/tcp	xenodochial_hoov

容器内部的 5000 端口映射到我们本地主机的 5000 端口上。

## 网络端口的快捷方式

通过 **docker ps** 命令可以查看到容器的端口映射，**docker** 还提供了另一个快捷方式 **docker port**，使用 **docker port** 可以查看指定（ID 或者名字）容器的某个确定端口映射到宿主机的端口号。

上面我们创建的 web 应用容器 ID 为 **bf08b7f2cd89** 名字为 **wizardly\_chandrasekhar**。

我可以使用 `docker port bf08b7f2cd89` 或 `docker port wizardly_chandrasekhar` 来查看容器端口的映射情况。

```
runoob@runoob:~$ docker port bf08b7f2cd89
```

```
5000/tcp -> 0.0.0.0:5000
```

```
runoob@runoob:~$ docker port wizardly_chandrasekhar
```

```
5000/tcp -> 0.0.0.0:5000
```

## 查看 WEB 应用程序日志

`docker logs` [ID或者名字] 可以查看容器内部的标准输出。

```
runoob@runoob:~$ docker logs -f bf08b7f2cd89
```

```
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

```
192.168.239.1 - - [09/May/2016 16:30:37] "GET / HTTP/1.1" 200 -
```

```
192.168.239.1 - - [09/May/2016 16:30:37] "GET /favicon.ico HTTP/1.1" 404 -
```

**-f**: 让 `docker logs` 像使用 `tail -f` 一样来输出容器内部的标准输出。

从上面，我们可以看到应用程序使用的是 `5000` 端口并且能够查看到应用程序的访问日志。

## 查看WEB应用程序容器的进程

我们还可以使用 `docker top` 来查看容器内部运行的进程

```
runoob@runoob:~$ docker top wizardly_chandrasekhar
```

UID	PID	PPID	...	TIME	CMD
root	23245	23228	...	00:00:00	python app.py

## 检查 WEB 应用程序

使用 `docker inspect` 来查看 Docker 的底层信息。它会返回一个 JSON 文件记录着 Docker 容器的配置和状态信息。

```
runoob@runoob:~$ docker inspect wizardly_chandrasekhar
```

```
[
  {
    "Id": "bf08b7f2cd897b5964943134aa6d373e355c286db9b9885b1f60b6e8f82b2b85",
    "Created": "2018-09-17T01:41:26.174228707Z",
    "Path": "python",
```

```
"Args": [

    "app.py"

],

"State": {

    "Status": "running",

    "Running": true,

    "Paused": false,

    "Restarting": false,

    "OOMKilled": false,

    "Dead": false,

    "Pid": 23245,

    "ExitCode": 0,

    "Error": "",

    "StartedAt": "2018-09-17T01:41:26.494185806Z",

    "FinishedAt": "0001-01-01T00:00:00Z"

},

.....
```

## 停止 **WEB** 应用容器

```
runoob@runoob:~$ docker stop wizardly_chandrasekhar

wizardly_chandrasekhar
```

## 重启 **WEB** 应用容器

已经停止的容器，我们可以使用命令 **docker start** 来启动。

```
runoob@runoob:~$ docker start wizardly_chandrasekhar

wizardly_chandrasekhar
```

**docker ps -l** 查询最后一次创建的容器：

```
# docker ps -l

CONTAINER ID        IMAGE               PORTS              NAMES
bf08b7f2cd89       training/webapp    0.0.0.0:5000->5000/tcp  wizardly_chandrasekhar
```

正在运行的容器，我们可以使用 `docker restart` 命令来重启

## 移除WEB应用容器

我们可以使用 `docker rm` 命令来删除不需要的容器

```
runoob@runoob:~$ docker rm wizardly_chandrasekhar

wizardly_chandrasekhar
```

删除容器时，容器必须是停止状态，否则会报如下错误

```
runoob@runoob:~$ docker rm wizardly_chandrasekhar

Error response from daemon: You cannot remove a running container bf08b7f2cd897b5964943134aa6d373e355c286db9b9885b1f60b6e8f82b2b85. Stop the container before attempting removal or force remove
```

[☐ Ubuntu Docker 安装](#)

[Docker 镜像使用](#) ☐

[☐ 点我分享笔记](#)

[反馈/建议](#)

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[☐ Docker 容器使用](#)

[Docker 容器连接](#) ☐

## Docker 镜像使用

当运行容器时，使用的镜像如果在本地中不存在，`docker` 就会自动从 `docker` 镜像仓库中下载，默认是从 `Docker Hub` 公共镜像源下载。

下面我们来学习：

- 1、管理和使用本地 `Docker` 主机镜像
- 2、创建镜像

## 列出镜像列表

我们可以使用 `docker images` 来列出本地主机上的镜像。

```
runoob@runoob:~$ docker images

REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              14.04              90d5884b1ee0       5 days ago         188 MB
```

php	5.6	f40e9e0f10c8	9 days ago	444.8 MB
nginx	latest	6f8d099c3adc	12 days ago	182.7 MB
mysql	5.6	f2e8d6c772c0	3 weeks ago	324.6 MB
httpd	latest	02ef73cf1bc0	3 weeks ago	194.4 MB
ubuntu	15.10	4e3b13c8a266	4 weeks ago	136.3 MB
hello-world	latest	690ed74de00f	6 months ago	960 B
training/webapp	latest	6fae60ef3446	11 months ago	348.8 MB

各个选项说明:

**REPOSITORY:** 表示镜像的仓库源

**TAG:** 镜像的标签

**IMAGE ID:** 镜像ID

**CREATED:** 镜像创建时间

**SIZE:** 镜像大小

同一仓库源可以有多个 **TAG**, 代表这个仓库源的不同个版本, 如**ubuntu**仓库源里, 有**15.10**、**14.04**等多个不同的版本, 我们使用 **REPOSITORY:TAG** 来定义不同的镜像。

所以, 我们如果要使用版本为**15.10**的**ubuntu**系统镜像来运行容器时, 命令如下:

```
runoob@runoob:~$ docker run -t -i ubuntu:15.10 /bin/bash

root@d77ccb2e5cca:/#
```

如果要使用版本为**14.04**的**ubuntu**系统镜像来运行容器时, 命令如下:

```
runoob@runoob:~$ docker run -t -i ubuntu:14.04 /bin/bash

root@39e968165990:/#
```

如果你不指定一个镜像的版本标签, 例如你只使用 **ubuntu**, **docker** 将默认使用 **ubuntu:latest** 镜像。

## 获取一个新的镜像

当我们在本地主机上使用一个不存在的镜像时 **Docker** 就会自动下载这个镜像。如果我们想预先下载这个镜像, 我们可以使用 **docker pull** 命令来下载它。

```
Crunoob@runoob:~$ docker pull ubuntu:13.10

13.10: Pulling from library/ubuntu

6599cada9f950: Pull complete

23eda618d451: Pull complete
```



```
f0be3084efe9: Pull complete

52de432f084b: Pull complete

a3ed95caeb02: Pull complete

Digest: sha256:15b79a6654811c8d992ebacdfbd5152fcf3d165e374e264076aa435214a947a3

Status: Downloaded newer image for ubuntu:13.10
```

下载完成后，我们可以直接使用这个镜像来运行容器。

## 查找镜像

我们可以从 **Docker Hub** 网站来搜索镜像，**Docker Hub** 网址为：<https://hub.docker.com/>

我们也可以使用 **docker search** 命令来搜索镜像。比如我们需要一个httpd的镜像来作为我们的web服务。我们可以通过 **docker search** 命令搜索 **httpd** 来寻找适合我们的镜像。

```
runoob@runoob:~$ docker search httpd
```

```
runoob@runoob:~$ docker search httpd
NAME                DESCRIPTION                STARS     OFFICIAL   AUTOMATED
httpd               The Apache HTTP Server Project 436       [OK]
centos/httpd        Docker image for Apache httpd with PHP 5 b... 1         [OK]
rgielen/httpd-image-php5  Httpd frontend allowing simple deployment ... 1         [OK]
graviteoio/httpd     httpd base image          0         [OK]
ibtech/httpd        Apache HTTPD padrão da IBTech 0         [OK]
rgielen/httpd-image-simple  Docker image for simple Apache httpd based... 0         [OK]
interferex/httpd      Simple HTTPD Server + TC Rate Limiting 0         [OK]
salison/httpd         Derivatives of the official httpd image 0         [OK]
sahrabkhan/httpd      Docker HTTPD + php5.6 (including php-mysql)... 0         [OK]
rgielen/httpd-image-drush  Apache HTTPD + Drupal Shell Docker image b... 0         [OK]
alizarion/httpd       httpd on centos with mod_auth_openid module 0         [OK]
jckorz/httpd-htaccess-ssl  Apache httpd with SSL and .htaccess files ... 0         [OK]
luisstanc/httpd        Apache http Server Docker image based on t... 0         [OK]
aintshvri/docker-httpd  Apache HTTPD Docker extension for legacy P... 0         [OK]
fenixsd/httpd         Apache HTTPD default for Fenix 0         [OK]
robertdebock/docker-rundock-httpd  A docker container running httpd to act as... 0         [OK]
chinakevnguoi/httpd     Automated build for httpd 0         [OK]
microwebapps/httpd-frontend-atomicapp  Httpd frontend Atomic App allowing simple ... 0         [OK]
jbot/httpd           Docker image with some httpd modules 0         [OK]
flem/httpd           Docker image with some httpd modules 0         [OK]
learninglayers/httpd    Docker image with some httpd modules 0         [OK]
steelorbis/httpd       local httpd 0         [OK]
efreson/httpd         A micro-sized httpd. Start serving files i... 0         [OK]
publici/httpd         httpd:latest 0         [OK]
```

**NAME:**镜像仓库源的名称

**DESCRIPTION:**镜像的描述

**OFFICIAL:**是否docker官方发布

## 拖取镜像

我们决定使用上图中的**httpd** 官方版本的镜像，使用命令 **docker pull** 来下载镜像。

```
runoob@runoob:~$ docker pull httpd

Using default tag: latest

latest: Pulling from library/httpd

8b87079b7a06: Pulling fs layer

a3ed95caeb02: Download complete

0d62ec9c6a76: Download complete

a329d50397b9: Download complete

ea7c1f032b5c: Waiting

be44112b72c7: Waiting
```

下载完成后，我们就可以使用这个镜像了。

```
runoob@runoob:~$ docker run httpd
```

## 创建镜像

当我们从docker镜像仓库中下载的镜像不能满足我们的需求时，我们可以通过以下两种方式对镜像进行更改。

- 1.从已经创建的容器中更新镜像，并且提交这个镜像
- 2.使用 **Dockerfile** 指令来创建一个新的镜像

## 更新镜像

更新镜像之前，我们需要使用镜像来创建一个容器。

```
runoob@runoob:~$ docker run -t -i ubuntu:15.10 /bin/bash
```

```
root@e218edb10161:/#
```

在运行的容器内使用 **apt-get update** 命令进行更新。

在完成操作之后，输入 **exit**命令来退出这个容器。

此时ID为**e218edb10161**的容器，是按我们的需求更改的容器。我们可以通过命令 **docker commit**来提交容器副本。

```
runoob@runoob:~$ docker commit -m="has update" -a="runoob" e218edb10161 runoob/ubuntu:v2
```

```
sha256:70bf1840fd7c0d2d8ef0a42a817eb29f854c1af8f7c59fc03ac7bdee9545aff8
```

各个参数说明：

**-m:**提交的描述信息

**-a:**指定镜像作者

**e218edb10161:** 容器ID

**runoob/ubuntu:v2:**指定要创建的目标镜像名

我们可以使用 **docker images** 命令来查看我们的新镜像 **runoob/ubuntu:v2:**

```
runoob@runoob:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
runoob/ubuntu	v2	70bf1840fd7c	15 seconds ago	158.5 MB
ubuntu	14.04	90d5884b1ee0	5 days ago	188 MB
php	5.6	f40e9e0f10c8	9 days ago	444.8 MB
nginx	latest	6f8d099c3adc	12 days ago	182.7 MB
mysql	5.6	f2e8d6c772c0	3 weeks ago	324.6 MB
httpd	latest	02ef73cf1bc0	3 weeks ago	194.4 MB

ubuntu	15.10	4e3b13c8a266	4 weeks ago	136.3 MB
hello-world	latest	690ed74de00f	6 months ago	960 B
training/webapp	latest	6fae60ef3446	12 months ago	348.8 MB

使用我们的新镜像 **runoob/ubuntu** 来启动一个容器

```
runoob@runoob:~$ docker run -t -i runoob/ubuntu:v2 /bin/bash

root@1a9fbdeb5da3:/#
```

## 构建镜像

我们使用命令 **docker build**，从零开始来创建一个新的镜像。为此，我们需要创建一个 **Dockerfile** 文件，其中包含一组指令来告诉 **Docker** 如何构建我们的镜像。

```
runoob@runoob:~$ cat Dockerfile

FROM      centos:6.7

MAINTAINER    Fisher "fisher@sudops.com"


RUN      /bin/echo 'root:123456' |chpasswd

RUN      useradd runoob

RUN      /bin/echo 'runoob:123456' |chpasswd

RUN      /bin/echo -e "LANG=\"en_US.UTF-8\"" >/etc/default/locale

EXPOSE    22

EXPOSE    80

CMD      /usr/sbin/sshd -D
```

每一个指令都会在镜像上创建一个新的层，每一个指令的前缀都必须是大写的。

第一条**FROM**，指定使用哪个镜像源

**RUN** 指令告诉**docker** 在镜像内执行命令，安装了什么。。。

然后，我们使用 **Dockerfile** 文件，通过 **docker build** 命令来构建一个镜像。

```
runoob@runoob:~$ docker build -t runoob/centos:6.7 .

Sending build context to Docker daemon 17.92 kB

Step 1 : FROM centos:6.7

---&gt; d95b5ca17cc3

Step 2 : MAINTAINER Fisher "fisher@sudops.com"

---&gt; Using cache
```

```
---&gt; 0c92299c6f03
```

```
Step 3 : RUN /bin/echo 'root:123456' |chpasswd
```

```
---&gt; Using cache
```

```
---&gt; 0397ce2fbd0a
```

```
Step 4 : RUN useradd runoob
```

```
.....
```

参数说明:

**-t**: 指定要创建的目标镜像名

**.**: Dockerfile 文件所在目录, 可以指定Dockerfile 的绝对路径

使用docker images 查看创建的镜像已经在列表中存在,镜像ID为860c279d2fec

```
runoob@runoob:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
runoob/centos	6.7	860c279d2fec	About a minute ago	190.6 MB
runoob/ubuntu	v2	70bf1840fd7c	17 hours ago	158.5 MB
ubuntu	14.04	90d5884b1ee0	6 days ago	188 MB
php	5.6	f40e9e0f10c8	10 days ago	444.8 MB
nginx	latest	6f8d099c3adc	12 days ago	182.7 MB
mysql	5.6	f2e8d6c772c0	3 weeks ago	324.6 MB
httpd	latest	02ef73cf1bc0	3 weeks ago	194.4 MB
ubuntu	15.10	4e3b13c8a266	5 weeks ago	136.3 MB
hello-world	latest	690ed74de00f	6 months ago	960 B
centos	6.7	d95b5ca17cc3	6 months ago	190.6 MB
training/webapp	latest	6fae60ef3446	12 months ago	348.8 MB

我们可以使用新的镜像来创建容器

```
runoob@runoob:~$ docker run -t -i runoob/centos:6.7 /bin/bash
```

```
[root@41c28d18b5fb /]# id runoob
```

```
uid=500(runoob) gid=500(runoob) groups=500(runoob)
```

从上面看到新镜像已经包含我们创建的用户runoob

## 设置镜像标签

我们可以使用 `docker tag` 命令，为镜像添加一个新的标签。

```
runoob@runoob:~$ docker tag 860c279d2fec runoob/centos:dev
```

`docker tag` 镜像ID，这里是 `860c279d2fec` ,用户名称、镜像源名(repository name)和新的标签名(tag)。  
使用 `docker images` 命令可以看到，ID为860c279d2fec的镜像多一个标签。

```
runoob@runoob:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
runoob/centos	6.7	860c279d2fec	5 hours ago	190.6 MB
runoob/centos	dev	860c279d2fec	5 hours ago	190.6 MB
runoob/ubuntu	v2	70bf1840fd7c	22 hours ago	158.5 MB
ubuntu	14.04	90d5884b1ee0	6 days ago	188 MB
php	5.6	f40e9e0f10c8	10 days ago	444.8 MB
nginx	latest	6f8d099c3adc	13 days ago	182.7 MB
mysql	5.6	f2e8d6c772c0	3 weeks ago	324.6 MB
httpd	latest	02ef73cf1bc0	3 weeks ago	194.4 MB
ubuntu	15.10	4e3b13c8a266	5 weeks ago	136.3 MB
hello-world	latest	690ed74de00f	6 months ago	960 B
centos	6.7	d95b5ca17cc3	6 months ago	190.6 MB
training/webapp	latest	6fae60ef3446	12 months ago	348.8 MB



# Docker 容器连接

前面我们实现了通过网络端口来访问运行在docker容器内的服务。下面我们来实现通过端口连接到一个docker容器

## 网络端口映射

我们创建了一个 python 应用的容器。

```
runoob@runoob:~$ docker run -d -P training/webapp python app.py

fce072cc88cee71b1cdceb57c2821d054a4a59f67da6b416fceb5593f059fc6d
```

另外，我们可以指定容器绑定的网络地址，比如绑定 127.0.0.1。  
我们使用 **-P** 参数创建一个容器，使用 **docker ps** 来看到端口5000绑定主机端口32768。

```
runoob@runoob:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	
NAMEs						
fce072cc88ce00/tcp	grave_hopper	training/webapp	"python app.py"	4 minutes ago	Up 4 minutes	0.0.0.0:32768->5000

我们也可以使用 **-p** 标识来指定容器端口绑定到主机端口。  
两种方式的区别是：

- P :是容器内部端口随机映射到主机的高端口。
- p : 是容器内部端口绑定到指定的主机端口。

```
runoob@runoob:~$ docker run -d -p 5000:5000 training/webapp python app.py

33e4523d30aaf0258915c368e66e03b49535de0ef20317d3f639d40222ba6bc0
```

```
runoob@runoob:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	
NAMEs						
33e4523d30aaf00/tcp	berserk_bartik	training/webapp	"python app.py"	About a minute ago	Up About a minute	0.0.0.0:5000->5000
fce072cc88ce00/tcp	grave_hopper	training/webapp	"python app.py"	8 minutes ago	Up 8 minutes	0.0.0.0:32768->5000

另外，我们可以指定容器绑定的网络地址，比如绑定127.0.0.1。

```
runoob@runoob:~$ docker run -d -p 127.0.0.1:5001:5000 training/webapp python app.py

95c6ceef88ca3e71eaf303c2833fd6701d8d1b2572b5613b5a932dfdf8a857c

runoob@runoob:~$ docker ps
```

CONTAINER ID NAMES	IMAGE	COMMAND	CREATED	STATUS	PORTS
95c6ceef88ca .1:5001->5000/tcp	training/webapp adoring_stonebraker	"python app.py"	6 seconds ago	Up 6 seconds	5000/tcp, 127.0.0
33e4523d30aa 0/tcp	training/webapp berserk_bartik	"python app.py"	3 minutes ago	Up 3 minutes	0.0.0.0:5000->500
fce072cc88ce 00/tcp	training/webapp grave_hopper	"python app.py"	10 minutes ago	Up 10 minutes	0.0.0.0:32768->50

这样我们就可以通过访问127.0.0.1:5001来访问容器的5000端口。

上面的例子中，默认都是绑定 tcp 端口，如果要绑定 UDP 端口，可以在端口后面加上 /udp。

```
runoob@runoob:~$ docker run -d -p 127.0.0.1:5000:5000/udp training/webapp python app.py

6779686f06f6204579c1d655dd8b2b31e8e809b245a97b2d3a8e35abe9dcd22a

runoob@runoob:~$ docker ps
```

CONTAINER ID NAMES	IMAGE	COMMAND	CREATED	STATUS	PORTS
6779686f06f6 .1:5000->5000/udp	training/webapp drunk_visvesvaraya	"python app.py"	4 seconds ago	Up 2 seconds	5000/tcp, 127.0.0
95c6ceef88ca .1:5001->5000/tcp	training/webapp adoring_stonebraker	"python app.py"	2 minutes ago	Up 2 minutes	5000/tcp, 127.0.0
33e4523d30aa 0/tcp	training/webapp berserk_bartik	"python app.py"	5 minutes ago	Up 5 minutes	0.0.0.0:5000->500
fce072cc88ce 00/tcp	training/webapp grave_hopper	"python app.py"	12 minutes ago	Up 12 minutes	0.0.0.0:32768->50

docker port 命令可以让我们快捷地查看端口的绑定情况。

```
runoob@runoob:~$ docker port adoring_stonebraker 5000

127.0.0.1:5001
```

## Docker容器连接

端口映射并不是唯一把 docker 连接到另一个容器的方法。

docker有一个连接系统允许将多个容器连接在一起，共享连接信息。

docker连接会创建一个父子关系，其中父容器可以看到子容器的信息。

## 容器命名

当我们创建一个容器的时候，docker会自动对它进行命名。另外，我们也可以使用--name标识来命名容器，例如：

```
runoob@runoob:~$ docker run -d -P --name runoob training/webapp python app.py

43780a6eabaaf14e590b6e849235c75f3012995403f97749775e38436db9a441
```

我们可以使用 **docker ps** 命令来查看容器名称。

```
runoob@runoob:~$ docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
43780a6eabaa	training/webapp	"python app.py"	3 minutes ago	Up 3 minutes	0.0.0.0:32769->5000/tcp

[❏ 点我分享笔记](#)

反馈/建议



# Docker 安装 Nginx

## 方法一、docker pull nginx(推荐)

查找 [Docker Hub](#) 上的 nginx 镜像

```
runoob@runoob:~/nginx$ docker search nginx
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
nginx	Official build of Nginx.	3260	[OK]	
jwilder/nginx-proxy	Automated Nginx reverse proxy for docker c...	674		[OK]
richarvey/nginx-php-fpm	Container running Nginx + PHP-FPM capable ...	207		[OK]
million12/nginx-php	Nginx + PHP-FPM 5.5, 5.6, 7.0 (NG), CentOS...	67		[OK]
maxexcloo/nginx-php	Docker framework container with Nginx and ...	57		[OK]
webdevops/php-nginx	Nginx with PHP-FPM	39		[OK]
h3nrik/nginx-ldap	NGINX web server with LDAP/AD, SSL and pro...	27		[OK]
bitnami/nginx	Bitnami nginx Docker Image	19		[OK]



```
maxexcloo/nginx          Docker framework container with Nginx inst...  7          [OK]

...
```

这里我们拉取官方的镜像

```
runoob@runoob:~/nginx$ docker pull nginx
```

等待下载完成后，我们就可以在本地镜像列表里查到 **REPOSITORY** 为 **nginx** 的镜像。

```
runoob@runoob:~/nginx$ docker images nginx
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	555bbd91e13c	3 days ago	182.8 MB

## 方法二、通过 **Dockerfile** 构建(不推荐)

### 创建 **Dockerfile**

首先，创建目录 **nginx**, 用于存放后面的相关东西。

```
runoob@runoob:~$ mkdir -p ~/nginx/www ~/nginx/logs ~/nginx/conf
```

**www**: 目录将映射为 **nginx** 容器配置的虚拟目录。

**logs**: 目录将映射为 **nginx** 容器的日志目录。

**conf**: 目录里的配置文件将映射为 **nginx** 容器的配置文件。

进入创建的 **nginx** 目录，创建 **Dockerfile** 文件，内容如下：

```
FROM debian:stretch-slim

LABEL maintainer="NGINX Docker Maintainers <docker-maint@nginx.com>"

ENV NGINX_VERSION 1.14.0-1~stretch

ENV NJS_VERSION 1.14.0.0.2.0-1~stretch

RUN set -x \

    && apt-get update \

    && apt-get install --no-install-recommends --no-install-suggests -y gnupg1 apt-transport-https ca-certificates \

    && \

    NGINX_GPGKEY=573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62; \

    found=''; \

    for server in \
```

```

ha.pool.sks-keyservers.net \

hkp://keyserver.ubuntu.com:80 \

hkp://p80.pool.sks-keyservers.net:80 \

pgp.mit.edu \

; do \

    echo "Fetching GPG key $NGINX_GPGKEY from $server"; \

    apt-key adv --keyserver "$server" --keyserver-options timeout=10 --recv-keys "$NGINX_GPGKEY" && found=yes &&
break; \

done; \

test -z "$found" && echo >&2 "error: failed to fetch GPG key $NGINX_GPGKEY" && exit 1; \

apt-get remove --purge --auto-remove -y gnupg1 && rm -rf /var/lib/apt/lists/* \

&& dpkgArch="$(dpkg --print-architecture)" \

&& nginxPackages=" \

    nginx=${NGINX_VERSION} \

    nginx-module-xslt=${NGINX_VERSION} \

    nginx-module-geoip=${NGINX_VERSION} \

    nginx-module-image-filter=${NGINX_VERSION} \

    nginx-module-njs=${NJS_VERSION} \

" \

&& case "$dpkgArch" in \

    amd64|i386) \

# arches officially built by upstream

        echo "deb https://nginx.org/packages/debian/ stretch nginx" >> /etc/apt/sources.list.d/nginx.list \

        && apt-get update \

        ;; \

    *) \

# we're on an architecture upstream doesn't officially build for

# let's build binaries from the published source packages

        echo "deb-src https://nginx.org/packages/debian/ stretch nginx" >> /etc/apt/sources.list.d/nginx.list \

        \

# new directory for storing sources and .deb files

        && tempDir="$(mktemp -d)" \

        && chmod 777 "$tempDir" \

```

```

# (777 to ensure APT's "_apt" user can access it too)

\

# save list of currently-installed packages so build dependencies can be cleanly removed later

&& savedAptMark="$(apt-mark showmanual)" \

\

# build .deb files from upstream's source packages (which are verified by apt-get)

&& apt-get update \

&& apt-get build-dep -y $nginxPackages \

&& ( \

    cd "$tempDir" \

    && DEB_BUILD_OPTIONS="nocheck parallel=$(nproc)" \

    apt-get source --compile $nginxPackages \

) \

# we don't remove APT lists here because they get re-downloaded and removed later

\

# reset apt-mark's "manual" list so that "purge --auto-remove" will remove all build dependencies

# (which is done after we install the built packages so we don't have to redownload any overlapping dependencies)

&& apt-mark showmanual | xargs apt-mark auto > /dev/null \

&& { [ -z "$savedAptMark" ] || apt-mark manual $savedAptMark; } \

\

# create a temporary local APT repo to install from (so that dependency resolution can be handled by APT, as it should be)

&& ls -lAFh "$tempDir" \

&& ( cd "$tempDir" && dpkg-scanpackages . > Packages ) \

&& grep '^Package: ' "$tempDir/Packages" \

&& echo "deb [ trusted=yes ] file://$tempDir ./" > /etc/apt/sources.list.d/temp.list \

# work around the following APT issue by using "Acquire::GzipIndexes=false" (overriding "/etc/apt/apt.conf.d/docker-gzip-indexes")

# Could not open file /var/lib/apt/lists/partial/_tmp_tmp.ODWljpQfkE_.Packages - open (13: Permission denied)

# ...

# E: Failed to fetch store:/var/lib/apt/lists/partial/_tmp_tmp.ODWljpQfkE_.Packages Could not open file /var/lib/apt/lists/partial/_tmp_tmp.ODWljpQfkE_.Packages - open (13: Permission denied)

&& apt-get -o Acquire::GzipIndexes=false update \

;; \

```

```

esac \

\

&& apt-get install --no-install-recommends --no-install-suggests -y \

    $nginxPackages \

    gettext-base \

    && apt-get remove --purge --auto-remove -y apt-transport-https ca-certificates && rm -rf /var/lib/apt/lists/* /et
c/apt/sources.list.d/nginx.list \

\

# if we have leftovers from building, let's purge them (including extra, unnecessary build deps)

&& if [ -n "$tempDir" ]; then \

    apt-get purge -y --auto-remove \

    && rm -rf "$tempDir" /etc/apt/sources.list.d/temp.list; \

fi

# forward request and error logs to docker log collector

RUN ln -sf /dev/stdout /var/log/nginx/access.log \

    && ln -sf /dev/stderr /var/log/nginx/error.log

EXPOSE 80

STOPSIGNAL SIGTERM

CMD ["nginx", "-g", "daemon off;"]

```

通过 **Dockerfile** 创建一个镜像，替换成你自己的名字。

```
docker build -t nginx .
```

创建完成后，我们可以在本地的镜像列表里查找到刚刚创建的镜像

```
runoob@runoob:~/nginx$ docker images nginx
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	555bbd91e13c	3 days ago	182.8 MB

## 使用 **nginx** 镜像

运行容器

```
runoob@runoob:~/nginx$ docker run -p 80:80 --name mynginx -v $PWD/www:/www -v $PWD/conf/nginx.conf:/etc/nginx/nginx.conf -v $PWD/logs:/wwwlogs -d nginx

45c89fab0bf9ad643bc7ab571f3ccd65379b844498f54a7c8a4e7ca1dc3a2c1e

runoob@runoob:~/nginx$
```

命令说明:

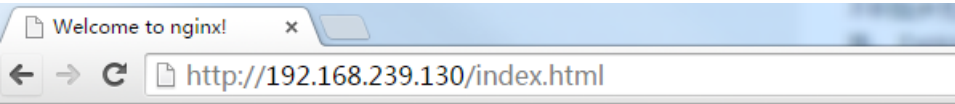
- p 80:80: 将容器的80端口映射到主机的80端口
- name mynginx: 将容器命名为mynginx
- v \$PWD/www:/www: 将主机中当前目录下的www挂载到容器的/www
- v \$PWD/conf/nginx.conf:/etc/nginx/nginx.conf: 将主机中当前目录下的nginx.conf挂载到容器的/etc/nginx/nginx.conf
- v \$PWD/logs:/wwwlogs: 将主机中当前目录下的logs挂载到容器的/wwwlogs

查看容器启动情况

```
runoob@runoob:~/nginx$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	PORTS	NAMES
45c89fab0bf9	nginx	"nginx -g 'daemon off'"	... 0.0.0.0:80->80/tcp, 443/tcp	mynginx
f2fa96138d71	tomcat	"catalina.sh run"	... 0.0.0.0:81->8080/tcp	tomcat

通过浏览器访问



Welcome to nginx!

☐ Docker 容器连接

Docker 安装 MySQL ☐

☐ 点我分享笔记

反馈/建议



# Docker 安装 PHP

## 安装 PHP 镜像

### 方法一、docker pull php

查找Docker Hub上的php镜像

```
runoob@runoob:~/php-fpm$ docker search php
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
php	While designed for web development, the PH...	1232	[OK]	
richarvey/nginx-php-fpm	Container running Nginx + PHP-FPM capable ...	207		[OK]
phpmyadmin/phpmyadmin	A web interface for MySQL and MariaDB.	123		[OK]
eboraas/apache-php	PHP5 on Apache (with SSL support), built o...	69		[OK]
php-zendserver	Zend Server - the integrated PHP applicati...	69	[OK]	
million12/nginx-php	Nginx + PHP-FPM 5.5, 5.6, 7.0 (NG), CentOS...	67		[OK]
webdevops/php-nginx	Nginx with PHP-FPM	39		[OK]
webdevops/php-apache	Apache with PHP-FPM (based on webdevops/php)	14		[OK]
phpunit/phpunit	PHPUnit is a programmer-oriented testing f...	14		[OK]
tetraweb/php	PHP 5.3, 5.4, 5.5, 5.6, 7.0 for CI and run...	12		[OK]
webdevops/php	PHP (FPM and CLI) service container	10		[OK]
...				

这里我们拉取官方的镜像,标签为5.6-fpm

```
runoob@runoob:~/php-fpm$ docker pull php:5.6-fpm
```

等待下载完成后,我们就可以在本地镜像列表里查到REPOSITORY为php,标签为5.6-fpm的镜像。

```
runoob@runoob:~/php-fpm$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
php	5.6-fpm	025041cd3aa5	6 days ago	456.3 MB

### 方法二、通过 Dockerfile 构建

创建Dockerfile

首先, 创建目录php-fpm,用于存放后面的相关东西。

```
runoob@runoob:~$ mkdir -p ~/php-fpm/logs ~/php-fpm/conf
```

logs目录将映射为php-fpm容器的日志目录

conf目录里的配置文件将映射为php-fpm容器的配置文件

进入创建的php-fpm目录，创建Dockerfile

```
FROM debian:jessie

# persistent / runtime deps

ENV PHPIZE_DEPS \

    autoconf \

    file \

    g++ \

    gcc \

    libc-dev \

    make \

    pkg-config \

    re2c

RUN apt-get update && apt-get install -y \

    $PHPIZE_DEPS \

    ca-certificates \

    curl \

    libedit2 \

    libsqlite3-0 \

    libxml2 \

    --no-install-recommends && rm -r /var/lib/apt/lists/*

ENV PHP_INI_DIR /usr/local/etc/php

RUN mkdir -p $PHP_INI_DIR/conf.d

##<autogenerated>##

ENV PHP_EXTRA_CONFIGURE_ARGS --enable-fpm --with-fpm-user=www-data --with-fpm-group=www-data

##</autogenerated>##

ENV GPG_KEYS 0BD78B5F97500D450838F95DFE857D9A90D90EC1 6E4F6AB321FDC07F2C332E3AC2BF0BC433CFC8B3
```

```
ENV PHP_VERSION 5.6.22

ENV PHP_FILENAME php-5.6.22.tar.xz

ENV PHP_SHA256 c96980d7de1d66c821a4ee5809df0076f925b2fe0b8c362d234d92f2f0a178e2


RUN set -xe \

    && buildDeps=" \

        $PHP_EXTRA_BUILD_DEPS \

        libcurl4-openssl-dev \

        libedit-dev \

        libsqlite3-dev \

        libssl-dev \

        libxml2-dev \

        xz-utils \

    " \

    && apt-get update && apt-get install -y $buildDeps --no-install-recommends && rm -rf /var/lib/apt/lists/* \

    && curl -fSL "http://php.net/get/$PHP_FILENAME/from/this/mirror" -o "$PHP_FILENAME" \

    && echo "$PHP_SHA256 *$PHP_FILENAME" | sha256sum -c - \

    && curl -fSL "http://php.net/get/$PHP_FILENAME.asc/from/this/mirror" -o "$PHP_FILENAME.asc" \

    && export GNUPGHOME="$(mktemp -d)" \

    && for key in $GPG_KEYS; do \

        gpg --keyserver ha.pool.sks-keyservers.net --recv-keys "$key"; \

    done \

    && gpg --batch --verify "$PHP_FILENAME.asc" "$PHP_FILENAME" \

    && rm -r "$GNUPGHOME" "$PHP_FILENAME.asc" \

    && mkdir -p /usr/src/php \

    && tar -xf "$PHP_FILENAME" -C /usr/src/php --strip-components=1 \

    && rm "$PHP_FILENAME" \

    && cd /usr/src/php \

    && ./configure \

        --with-config-file-path="$PHP_INI_DIR" \

        --with-config-file-scan-dir="$PHP_INI_DIR/conf.d" \

        $PHP_EXTRA_CONFIGURE_ARGS \
```



```

--disable-cgi \

# --enable-mysqld is included here because it's harder to compile after the fact than extensions are (since it's a p
lugin for several extensions, not an extension in itself)

--enable-mysqld \

# --enable-mbstring is included here because otherwise there's no way to get pecl to use it properly (see https://git
hub.com/docker-library/php/issues/195)

--enable-mbstring \

--with-curl \

--with-libedit \

--with-openssl \

--with-zlib \

&& make -j"${nproc}" \

&& make install \

&& { find /usr/local/bin /usr/local/sbin -type f -executable -exec strip --strip-all '{}' + || true; } \

&& make clean \

&& apt-get purge -y --auto-remove -o APT::AutoRemove::RecommendsImportant=false -o APT::AutoRemove::SuggestsImpor
tant=false $buildDeps

COPY docker-php-ext-* /usr/local/bin/

##<autogenerated>##

WORKDIR /var/www/html

RUN set -ex \

&& cd /usr/local/etc \

&& if [ -d php-fpm.d ]; then \

    # for some reason, upstream's php-fpm.conf.default has "include=NONE/etc/php-fpm.d/*.conf"

    sed 's!=NONE/!=!g' php-fpm.conf.default | tee php-fpm.conf > /dev/null; \

    cp php-fpm.d/www.conf.default php-fpm.d/www.conf; \

else \

    # PHP 5.x don't use "include=" by default, so we'll create our own simple config that mimics PHP 7+ for consi
stency

    mkdir php-fpm.d; \

    cp php-fpm.conf.default php-fpm.d/www.conf; \

{ \

```

```

        echo '[global]'; \

        echo 'include=etc/php-fpm.d/*.conf'; \

    } | tee php-fpm.conf; \

fi \

&& { \

    echo '[global]'; \

    echo 'error_log = /proc/self/fd/2'; \

    echo; \

    echo '[www]'; \

    echo '; if we send this to /proc/self/fd/1, it never appears'; \

    echo 'access.log = /proc/self/fd/2'; \

    echo; \

    echo 'clear_env = no'; \

    echo; \

    echo '; Ensure worker stdout and stderr are sent to the main error log.'; \

    echo 'catch_workers_output = yes'; \

} | tee php-fpm.d/docker.conf \

&& { \

    echo '[global]'; \

    echo 'daemonize = no'; \

    echo; \

    echo '[www]'; \

    echo 'listen = [::]:9000'; \

} | tee php-fpm.d/zz-docker.conf

```

EXPOSE 9000

CMD ["php-fpm"]

通过Dockerfile创建一个镜像，替换成你自己的名字

```
runoob@runoob:~/php-fpm$ docker build -t php:5.6-fpm .
```

创建完成后，我们可以在本地的镜像列表里查找到刚刚创建的镜像

```
runoob@runoob:~/php-fpm$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
php	5.6-fpm	025041cd3aa5	6 days ago	456.3 MB

## 使用php-fpm镜像

### 运行容器

```
runoob@runoob:~/php-fpm$ docker run -p 9000:9000 --name myphp-fpm -v ~/nginx/www:/www -v $PWD/conf:/usr/local/etc/php -v $PWD/logs:/phplogs -d php:5.6-fpm
```

```
00c5aa4c2f93ec3486936f45b5f2b450187a9d09acb18f5ac9aa7a5f405dbedf
```

```
runoob@runoob:~/php-fpm$
```

命令说明:

-p 9000:9000 :将容器的9000端口映射到主机的9000端口

--name myphp-fpm :将容器命名为myphp-fpm

-v ~/nginx/www:/www :将主机中项目的目录www挂载到容器的/www

-v \$PWD/conf:/usr/local/etc/php : 将主机中当前目录下的conf目录挂载到容器的/usr/local/etc/php

-v \$PWD/logs:/phplogs : 将主机中当前目录下的logs目录挂载到容器的/phplogs

### 查看容器启动情况


```
runoob@runoob:~/php-fpm$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	...	PORTS	NAMES
00c5aa4c2f93	php:5.6-fpm	"php-fpm"	...	0.0.0.0:9000->9000/tcp	myphp-fpm

通过浏览器访问phpinfo()

192.168.239.130/index.php

PHP Version 5.6.22



System	Linux 00c5aa4c2f93 4.2.0-16-generic #19-Ubuntu SMP Thu Oct 8 15:35:06 UTC 2015 x86_64
Build Date	Jun 10 2016 03:18:39
Configure Command	'./configure' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' '--disable-cgi' '--enable-mysqlnd' '--enable-mbstring' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib'
Server API	FFPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	/usr/local/etc/php/php.ini

PS:此处是通过nginx+php实现web服务，nginx配置文件的fastcgi\_pass应该配置为myphp-fpm容器的IP。

```
fastcgi_pass 172.17.0.4:9000;
```

容器IP的查方法

```
docker inspect 容器ID或容器名 |grep '"IPAddress"'
```

☐ Docker 安装 MySQL

Docker 安装 Tomcat ☐



1 篇笔记  
#1

☐ 写笔记



Docker 配置 nginx、php-fpm、mysql  
运行环境



创建目录

```
mkdir -p /Users/sui/docker/nginx/conf.d && mkdir /Users/sui/www && cd /Users/sui/docker/nginx/conf.d && sudo touch default.conf
```

启动 php-fpm

解释执行 php 需要 php-fpm，先让它运行起来：

```
docker run --name sui-php -d \

-v /Users/sui/www:/var/www/html:ro \

php:7.1-fpm
```

`--name sui-php` 是容器的名字。

`/Users/sui/www` 是本地 php 文件的存储目录，`/var/www/html` 是容器内 php 文件的存储目录，`ro` 表示只读。

编辑 nginx 配置文件

配置文件位置：`/Users/sui/docker/nginx/conf.d/default.conf`。

```
server {

    listen      80;

    server_name localhost;

    location / {

        root    /usr/share/nginx/html;

        index   index.html index.htm;
```

```

}

error_page    500 502 503 504    /50x.html;

location = /50x.html {

    root       /usr/share/nginx/html;

}

location ~ /\.php$ {

    fastcgi_pass    php:9000;

    fastcgi_index   index.php;

    fastcgi_param   SCRIPT_FILENAME    /var/www/html/$fastcgi_script_name;

    include          fastcgi_params;

}

}

```

说明：

php:9000 表示 php-fpm 服务的访问路径，下文还会提及。

/var/www/html 是 **sui-php** 中 php 文件的存储路径，经 docker 映射，变成本地路径 /Users/sui/www（可以再看一眼 php-fpm 启动命令启动 nginx

```

docker run --name sui-nginx -p 80:80 -d \

    -v /Users/sui/www:/usr/share/nginx/html:ro \

    -v /Users/sui/docker/nginx/conf.d:/etc/nginx/conf.d:ro \

    --link sui-php:php \

    nginx

```

-p 80:80 用于添加端口映射，把 **sui-nginx** 中的 80 端口暴露出来。

/Users/sui/www 是本地 html 文件的存储目录，/usr/share/nginx/html 是容器内 html 文件的存储目录。

/Users/sui/docker/nginx/conf.d 是本地 nginx 配置文件的存储目录，/etc/nginx/conf.d 是容器内 nginx 配置文件的存储目录。

--link sui-php:php 把 **sui-php** 的网络并入 **sui-nginx**。并通过修改 **sui-nginx** 的 /etc/hosts，把域名 **php** 映射成 127.0.0.1，让 nginx 通过 php:9000 访问 php-fpm。

```
1. theo@Theo-iMac: /Users/sui/docker/nginx/conf.d (zsh)
cd: not a directory: /Users/sui/docker/nginx/conf.d/default.conf
→ sui cd /Users/sui/docker/nginx/conf.d/
→ conf.d ls
default.conf
→ conf.d sudo vim default.conf
Password:
→ conf.d docker run --name sui-nginx -p 80:80 -d \
> -v /Users/sui/www:/usr/share/nginx/html:ro \
> -v /Users/sui/docker/nginx/conf.d:/etc/nginx/conf.d:ro \
> --link sui-php:php \
> nginx
docker: Error response from daemon: Conflict. The container name "/sui-nginx" is
already in use by container "d6b1647fcf274a5a21e01ed6fa255a3f548a9c2fe9123a542f
151e773ff5717d". You have to remove (or rename) that container to be able to reu
se that name.
See 'docker run --help'.
→ conf.d docker rm -f sui-nginx
sui-nginx
→ conf.d docker run --name sui-nginx -p 80:80 -d \
-v /Users/sui/www:/usr/share/nginx/html:ro \
-v /Users/sui/docker/nginx/conf.d:/etc/nginx/conf.d:ro \
--link sui-php:php \
nginx
9e57b41ce4111012efe9e9ff4611a87d66c838fcd6938675ae509d75f606971e
→ conf.d
```

测试结果

在 /Users/sui/www 下放两个文件: index.html index.php

```
1. theo@Theo-iMac: /Users/sui/www (zsh)
already in use by container "d6b1647fcf274a5a21e01ed6fa255a3f548a9c2fe9123a542f
151e773ff5717d". You have to remove (or rename) that container to be able to reu
se that name.
See 'docker run --help'.
→ conf.d docker rm -f sui-nginx
sui-nginx
→ conf.d docker run --name sui-nginx -p 80:80 -d \
-v /Users/sui/www:/usr/share/nginx/html:ro \
-v /Users/sui/docker/nginx/conf.d:/etc/nginx/conf.d:ro \
--link sui-php:php \
nginx
9e57b41ce4111012efe9e9ff4611a87d66c838fcd6938675ae509d75f606971e
→ conf.d ls
default.conf
→ conf.d cd /Users/sui/www
→ www ls
→ www vim index.html
→ www sudo vim index.html
Password:
→ www sudo vim index.php
→ www curl localhost/index.html
6666666
→ www curl localhost/index.php
string:
→ www
```

mysql 和 phpmyadmin

mysql 服务器

```
sudo mkdir -p /Users/sui/docker/mysql/data /Users/sui/docker/mysql/logs /Users/sui/docker/mysql/conf
```

data 目录将映射为 mysql 容器配置的数据文件存放路径

logs 目录将映射为 mysql 容器的日志目录

conf 目录里的配置文件将映射为 mysql 容器的配置文件

```
docker run -p 3307:3306 --name sui-mysql -v /Users/sui/docker/mysql/conf:/etc/mysql -v /Users/sui/docker/mysql/logs:/logs
-v /Users/sui/docker/mysql/data:/mysql_data -e MYSQL_ROOT_PASSWORD=123456 -d --link sui-php mysql
```

进入mysql客户端:

```
docker run -it --link sui-mysql:mysql --rm mysql sh -c 'exec mysql -h"$MYSQL_PORT_3306_TCP_ADDR" -P"$MYSQL_PORT_3306_TCP_P
ORT" -uroot -p"$MYSQL_ENV_MYSQL_ROOT_PASSWORD'
```

注意：我本地 3306 端口有 mysql, 所以这里用3307端口。

```
1. theo@Theo-iMac: /Users/sui/www (zsh)
9e57b41ce411      nginx      "nginx -g 'daemon ..."   About an hour ago   Up About an hour      0.0.0.0:80->80/tcp
sui-nginx
9e9aa37562e0      php:7.1-fpm  "docker-php-entryp..."   About an hour ago   Up About an hour      9000/tcp
sui-php
→ www curl localhost:3307
5.7.18ziY'80y000;}*)aB(v0x_mysql_native_password000ot packets out of order
→ www docker run -it --link sui-mysql:mysql --rm mysql sh -c 'exec mysql -h"$MYSQL_PORT_3306_TCP_ADDR" -P"$MYSQL_PORT_3306_TCP_PORT" -uroot -p"$MYSQL_ENV_MYSQL_ROOT_PASSWORD"'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 5.7.18 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
+-----+
```

phpmyadmin

```
docker run --name sui-myadmin -d --link sui-mysql:db -p 8080:80 phpmyadmin/phpmyadmin
```

```
1. theo@Theo-iMac: /Users/sui/www (zsh)
      NAMES
8f851ca9ed32      mysql      "docker-entrypoint..."   17 minutes ago     Up 17 minutes        0.0.0.0:3307->3306/
tcp sui-mysql
9e57b41ce411      nginx      "nginx -g 'daemon ..."   About an hour ago   Up About an hour      0.0.0.0:80->80/tcp
sui-nginx
9e9aa37562e0      php:7.1-fpm  "docker-php-entryp..."   About an hour ago   Up About an hour      9000/tcp
sui-php
→ www docker rm -f sui-myadmin
sui-myadmin
→ www docker run --name sui-myadmin -d --link sui-mysql:db -p 8080:80 phpmyadmin/phpmyadmin
df34e188c3b7943203b78dc206c24e05d9ac766cad02346ca1a88b6398cb0f65
→ www docker rm -f sui-myadmin
sui-myadmin
→ www docker run --name sui-myadmin -d --link sui-mysql:db -p 80:80 phpmyadmin/phpmyadmin
4f564b19cace9f436b44fadd7de8d03736471b1ac8f730127a7b4dbfa77f7057
docker: Error response from daemon: driver failed programming external connectivity on endpoint sui-myadmin (253d7c796982771f403c6b1dc6afe0713f58dd11cf19ad34e064d63d5df1d3d3): Bind for 0.0.0.0:80 failed: port is already allocated.
→ www docker run --name sui-myadmin -d --link sui-mysql:db -p 8080:80 phpmyadmin/phpmyadmin
docker: Error response from daemon: Conflict. The container name "/sui-myadmin" is already in use by container "4f564b19cace9f436b44fadd7de8d03736471b1ac8f730127a7b4dbfa77f7057". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
→ www docker rm -f sui-myadmin
sui-myadmin
→ www docker run --name sui-myadmin -d --link sui-mysql:db -p 8080:80 phpmyadmin/phpmyadmin
5b20743f7db57324cd663c5d2a80ff41ca337399960f437e30f54ce4b8f8ed10
→ www
```

大功告成:

localhost:8080/sql.php?server=1&db=sys&table=sys\_config&pos=0

应用 admin.verybeaut.com CodeIgniter 用户指... 朽木博客 | 关注互联... free-programming-... Nginx 使用手册\_w3c... MySQL联接查询操作...

phpMyAdmin

近期访问 表收藏夹

新建  
information\_schema  
mysql  
performance\_schema  
sys

服务器: db 数据库: sys 表: sys\_config

浏览 结构 SQL 搜索 导出

⚠ Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

正在显示第 0 - 5 行 (共 6 行, 查询花费 0.0044 秒。)

SELECT \* FROM `sys\_config`

☐ 性能分析 [ 编辑内嵌 ] [ 编辑 ] [ 解析 SQL ] [ 创建 PHP 代码 ] [ 刷新 ]

☐ 显示全部 行数: 25 过滤行: 在表中搜索 按索引排序: 无

+ 选项

variable	value	set_time	set_by
diagnostics.allow_i_s_tables	OFF	2017-07-15 06:13:58	NULL
diagnostics.include_raw	OFF	2017-07-15 06:13:58	NULL
ps_thread_trx_info.max_length	65535	2017-07-15 06:13:58	NULL
statement_performance_analyzer.limit	100	2017-07-15 06:13:58	NULL
statement_performance_analyzer.view	NULL	2017-07-15 06:13:58	NULL
statement_truncate_len	64	2017-07-15 06:13:58	NULL

☐ 显示全部 行数: 25 过滤行: 在表中搜索 按索引排序: 无

查询结果操作

pengqiangsheng1个月前 (08-31)

反馈/建议

Copyright © 2013-2018 菜鸟教程 runoob.com All Rights Reserved. 备案号: 闽ICP备15012807号-1



首页 HTML CSS JS 本地书签

☐ Docker 安装 Nginx

Docker 安装 PHP ☐

# Docker 安装 MySQL

## 方法一、docker pull mysql

查找Docker Hub上的mysql镜像

```
runoob@runoob:/mysql$ docker search mysql
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
mysql	MySQL is a widely used, open-source relati...	2529	[OK]	
mysql/mysql-server	Optimized MySQL Server Docker images. Crea...	161		[OK]
centurylink/mysql	Image containing mysql. Optimized to be li...	45		[OK]
sameersbn/mysql		36		[OK]



google/mysql	MySQL server for Google Compute Engine	16	[OK]
appcontainers/mysql	Centos/Debian Based Customizable MySQL Con...	8	[OK]
marvambass/mysql	MySQL Server based on Ubuntu 14.04	6	[OK]
drupaldocker/mysql	MySQL for Drupal	2	[OK]
azukiapp/mysql	Docker image to run MySQL by Azuki - http:...	2	[OK]
...			

这里我们拉取官方的镜像,标签为5.6

```
runoob@runoob:~/mysql$ docker pull mysql:5.6
```

等待下载完成后,我们就可以在本地镜像列表里查到REPOSITORY为mysql,标签为5.6的镜像。

```
runoob@runoob:~/mysql$ docker images |grep mysql
```

mysql	5.6	2c0964ec182a	3 weeks ago	329 MB
-------	-----	--------------	-------------	--------

## 方法二、通过 Dockerfile构建

创建Dockerfile

首先,创建目录mysql,用于存放后面的相关东西。

```
runoob@runoob:~$ mkdir -p ~/mysql/data ~/mysql/logs ~/mysql/conf
```

data目录将映射为mysql容器配置的数据文件存放路径

logs目录将映射为mysql容器的日志目录

conf目录里的配置文件将映射为mysql容器的配置文件

进入创建的mysql目录,创建Dockerfile

```
FROM debian:jessie

# add our user and group first to make sure their IDs get assigned consistently, regardless of whatever dependencies
get added

RUN groupadd -r mysql && useradd -r -g mysql mysql

# add gosu for easy step-down from root

ENV GOSU_VERSION 1.7

RUN set -x \

    && apt-get update && apt-get install -y --no-install-recommends ca-certificates wget && rm -rf /var/lib/apt/lists
/* \

    && wget -O /usr/local/bin/gosu "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$(dpkg --prin
t-architecture)" \
```

```

    && wget -O /usr/local/bin/gosu.asc "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$(dpkg --
print-architecture).asc" \

    && export GNUPGHOME="$(mktemp -d)" \

    && gpg --keyserver ha.pool.sks-keyservers.net --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4 \

    && gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu \

    && rm -r "$GNUPGHOME" /usr/local/bin/gosu.asc \

    && chmod +x /usr/local/bin/gosu \

    && gosu nobody true \

    && apt-get purge -y --auto-remove ca-certificates wget

RUN mkdir /docker-entrypoint-initdb.d

# FATAL ERROR: please install the following Perl modules before executing /usr/local/mysql/scripts/mysql_install_db:

# File::Basename

# File::Copy

# Sys::Hostname

# Data::Dumper

RUN apt-get update && apt-get install -y perl pwgen --no-install-recommends && rm -rf /var/lib/apt/lists/*

# gpg: key 5072E1F5: public key "MySQL Release Engineering <mysql-build@oss.oracle.com>" imported

RUN apt-key adv --keyserver ha.pool.sks-keyservers.net --recv-keys A4A9406876FCBD3C456770C88C718D3B5072E1F5

ENV MYSQL_MAJOR 5.6

ENV MYSQL_VERSION 5.6.31-1debian8

RUN echo "deb http://repo.mysql.com/apt/debian/ jessie mysql-${MYSQL_MAJOR}" > /etc/apt/sources.list.d/mysql.list

# the "/var/lib/mysql" stuff here is because the mysql-server postinst doesn't have an explicit way to disable the my
sql_install_db codepath besides having a database already "configured" (ie, stuff in /var/lib/mysql/mysql)

# also, we set debconf keys to make APT a little quieter

RUN { \

    echo mysql-community-server mysql-community-server/data-dir select ''; \

    echo mysql-community-server mysql-community-server/root-pass password ''; \

```

```

    echo mysql-community-server mysql-community-server/re-root-pass password ''; \

    echo mysql-community-server mysql-community-server/remove-test-db select false; \

} | debconf-set-selections \

&& apt-get update && apt-get install -y mysql-server="${MYSQL_VERSION}" && rm -rf /var/lib/apt/lists/* \

&& rm -rf /var/lib/mysql && mkdir -p /var/lib/mysql /var/run/mysqld \

&& chown -R mysql:mysql /var/lib/mysql /var/run/mysqld \

# ensure that /var/run/mysqld (used for socket and lock files) is writable regardless of the UID our mysqld instance
ends up having at runtime

&& chmod 777 /var/run/mysqld

# comment out a few problematic configuration values

# don't reverse lookup hostnames, they are usually another container

RUN sed -Ei 's/^(bind-address|log)/#&/' /etc/mysql/my.cnf \

    && echo 'skip-host-cache\nskip-name-resolve' | awk '{ print } $1 == "[mysqld]" && c == 0 { c = 1; system("cat") }'
    /etc/mysql/my.cnf > /tmp/my.cnf \

    && mv /tmp/my.cnf /etc/mysql/my.cnf

VOLUME /var/lib/mysql

COPY docker-entrypoint.sh /usr/local/bin/

RUN ln -s usr/local/bin/docker-entrypoint.sh /entrypoint.sh # backwards compat

ENTRYPOINT ["docker-entrypoint.sh"]

EXPOSE 3306

CMD ["mysqld"]

```

通过Dockerfile创建一个镜像，替换成你自己的名字

```
runoob@runoob:~/mysql$ docker build -t mysql .
```

创建完成后，我们可以在本地的镜像列表里查找到刚刚创建的镜像

```
runoob@runoob:~/mysql$ docker images |grep mysql
```

mysql	5.6	2c0964ec182a	3 weeks ago	329 MB
-------	-----	--------------	-------------	--------

## 使用mysql镜像

运行容器

```
runoob@runoob:~/mysql$ docker run -p 3306:3306 --name mymysql -v $PWD/conf:/etc/mysql/conf.d -v $PWD/logs:/logs -v $PWD/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=123456 -d mysql:5.6

21cb89213c93d805c5bacf1028a0da7b5c5852761ba81327e6b99bb3ea89930e

runoob@runoob:~/mysql$
```

命令说明：

- p 3306:3306: 将容器的 3306 端口映射到主机的 3306 端口。
- v -v \$PWD/conf:/etc/mysql/conf.d: 将主机当前目录下的 conf/my.cnf 挂载到容器的 /etc/mysql/my.cnf。
- v \$PWD/logs:/logs: 将主机当前目录下的 logs 目录挂载到容器的 /logs。
- v \$PWD/data:/var/lib/mysql : 将主机当前目录下的data目录挂载到容器的 /var/lib/mysql 。
- e MYSQL\_ROOT\_PASSWORD=123456: 初始化 root 用户的密码。

查看容器启动情况

```
runoob@runoob:~/mysql$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	...	PORTS	NAMES
21cb89213c93	mysql:5.6	"docker-entrypoint.sh"	...	0.0.0.0:3306->3306/tcp	mymysql

❏

2 篇笔记

#2

❏

写笔记

最新官方MySQL(5.7.19)的docker镜像在创建时映射的配置文件目录有所不同，在此记录并分享给大家：  
官方原文：  
The MySQL startup configuration is specified in the file /etc/mysql/my.cnf, and that file in turn includes any files found in the /etc/mysql/conf.d directory that end with .cnf. Settings in files in this directory will augment and/or override settings in /etc/mysql/my.cnf. If you want to use a customized MySQL configuration, you can create your alternative configuration file in a directory on the host machine and then mount that directory location as /etc/mysql/conf.d inside the mysql container.  
大概意思是说：  
MySQL(5.7.19)的默认配置文件是 /etc/mysql/my.cnf 文件。如果想要自定义配置，建议向 /etc/mysql/conf.d 目录中创建 .cnf 文件。新建的文件可以任意起名，只要保证后缀名是 cnf 即可。新建的文件中的配置项可以覆盖 /etc/mysql/my.cnf 中的配置项。  
具体操作：  
首先需要创建将要映射到容器中的目录以及.cnf文件，然后再创建容器

```
# pwd

/opt

# mkdir -p docker_v/mysql/conf

# cd docker_v/mysql/conf

# touch my.cnf

# docker run -p 3306:3306 --name mysql -v /opt/docker_v/mysql/conf:/etc/mysql/conf.d -e MYSQL_ROOT_PASSWORD=123456 -d imageID
```

4ec4f56455ea2d6d7251a05b7f308e314051fdad2c26bf3d0f27a9b0c0a71414

命令说明:  
-p 3306:3306: 将容器的3306端口映射到主机的3306端口  
-v /opt/docker\_v/mysql/conf/etc/mysql/conf.d: 将主机/opt/docker\_v/mysql/conf目录挂载到容器的/etc/mysql/conf.d  
-e MYSQL\_ROOT\_PASSWORD=123456: 初始化root用户的密码  
-d: 后台运行容器, 并返回容器ID  
imageID: mysql镜像ID  
查看容器运行情况

```
# docker ps
```

CONTAINER ID	IMAGE	COMMAND	... PORTS	NAMES
4ec4f56455ea	c73c7527c03a	"docker-entrypoint.sh"	... 0.0.0.0:3306->3306/tcp	mysql

Brian1年前 (2017-09-08)  
#1



docker 安装 mysql 8 版本

```
# docker 中下载 mysql

docker pull mysql

#启动

docker run --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=Lzsl0v123! -d mysql

#进入容器

docker exec -it mysql bash

#登录mysql

mysql -u root -p

ALTER USER 'root'@'localhost' IDENTIFIED BY 'Lzsl0v123!';

#添加远程登录用户

CREATE USER 'liaozesong'@'%' IDENTIFIED WITH mysql_native_password BY 'Lzsl0v123!';

GRANT ALL PRIVILEGES ON *.* TO 'liaozesong'@'%';
```

liaozesong2个月前 (07-30)

反馈/建议



# Docker 安装 Tomcat

## 方法一、docker pull tomcat

查找Docker Hub上的tomcat镜像

```
runoob@runoob:~/tomcat$ docker search tomcat
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
tomcat	Apache Tomcat is an open source implementa...	744	[OK]	
dordoka/tomcat	Ubuntu 14.04, Oracle JDK 8 and Tomcat 8 ba...	19		[OK]
consol/tomcat-7.0	Tomcat 7.0.57, 8080, "admin/admin"	16		[OK]
consol/tomcat-8.0	Tomcat 8.0.15, 8080, "admin/admin"	14		[OK]
cloudesire/tomcat	Tomcat server, 6/7/8	8		[OK]
davidcaste/alpine-tomcat	Apache Tomcat 7/8 using Oracle Java 7/8 wi...	6		[OK]
andreptb/tomcat	Debian Jessie based image with Apache Tomc...	4		[OK]
kieker/tomcat		2		[OK]
fbrx/tomcat	Minimal Tomcat image based on Alpine Linux	2		[OK]
jtech/tomcat	Latest Tomcat production distribution on l...	1		[OK]

这里我们拉取官方的镜像

```
runoob@runoob:~/tomcat$ docker pull tomcat
```

等待下载完成后，我们就可以在本地镜像列表里查到REPOSITORY为tomcat的镜像。

```
runoob@runoob:~/tomcat$ docker images|grep tomcat
```

tomcat	latest	70f819d3d2d9	7 days ago	335.8 MB
--------	--------	--------------	------------	----------

## 方法二、通过 Dockerfile 构建

创建Dockerfile

首先，创建目录tomcat,用于存放后面的相关东西。

```
runoob@runoob:~$ mkdir -p ~/tomcat/webapps ~/tomcat/logs ~/tomcat/conf
```

webapps目录将映射为tomcat容器配置的应用程序目录

logs目录将映射为tomcat容器的日志目录

conf目录里的配置文件将映射为tomcat容器的配置文件

进入创建的tomcat目录，创建Dockerfile

```
FROM openjdk:8-jre

ENV CATALINA_HOME /usr/local/tomcat

ENV PATH $CATALINA_HOME/bin:$PATH

RUN mkdir -p "$CATALINA_HOME"

WORKDIR $CATALINA_HOME

# let "Tomcat Native" live somewhere isolated

ENV TOMCAT_NATIVE_LIBDIR $CATALINA_HOME/native-jni-lib

ENV LD_LIBRARY_PATH ${LD_LIBRARY_PATH:+$LD_LIBRARY_PATH:$TOMCAT_NATIVE_LIBDIR}

# runtime dependencies for Tomcat Native Libraries

# Tomcat Native 1.2+ requires a newer version of OpenSSL than debian:jessie has available

# > checking OpenSSL library version >= 1.0.2...

# > configure: error: Your version of OpenSSL is not compatible with this version of tcnative

# see http://tomcat.10.x6.nabble.com/VOTE-Release-Apache-Tomcat-8-0-32-tp5046007p5046024.html (and following discussion)

# and https://github.com/docker-library/tomcat/pull/31

ENV OPENSSL_VERSION 1.1.0f-3+deb9u2

RUN set -ex; \

    currentVersion="$(dpkg-query --show --showformat '${Version}\n' openssl)"; \

    if dpkg --compare-versions "$currentVersion" '<<' "$OPENSSL_VERSION"; then \

        if ! grep -q stretch /etc/apt/sources.list; then \

# only add stretch if we're not already building from within stretch

            { \

                echo 'deb http://deb.debian.org/debian stretch main'; \

                echo 'deb http://security.debian.org stretch/updates main'; \

                echo 'deb http://deb.debian.org/debian stretch-updates main'; \

            } > /etc/apt/sources.list.d/stretch.list; \

        { \

# add a negative "Pin-Priority" so that we never ever get packages from stretch unless we explicitly request them
```

```

        echo 'Package: *'; \

        echo 'Pin: release n=stretch*'; \

        echo 'Pin-Priority: -10'; \

        echo; \

# ... except OpenSSL, which is the reason we're here

        echo 'Package: openssl libssl*'; \

        echo "Pin: version $OPENSSL_VERSION"; \

        echo 'Pin-Priority: 990'; \

    } > /etc/apt/preferences.d/stretch-openssl; \

fi; \

apt-get update; \

apt-get install -y --no-install-recommends openssl="$OPENSSL_VERSION"; \

rm -rf /var/lib/apt/lists/*; \

fi

RUN apt-get update && apt-get install -y --no-install-recommends \

    libapr1 \

    && rm -rf /var/lib/apt/lists/*

# see https://www.apache.org/dist/tomcat/tomcat-$TOMCAT_MAJOR/KEYS

# see also "update.sh" (https://github.com/docker-library/tomcat/blob/master/update.sh)

ENV GPG_KEYS 05AB33110949707C93A279E3D3EFE6B686867BA6 07E48665A34DCAFAE522E5E6266191C37C037D42 47309207D818FFD8DCD3F8
3F1931D684307A10A5 541FBE7D8F78B25E055DDEE13C370389288584E7 61B832AC2F1C5A90F0F9B00A1C506407564C17A3 713DA88BE5091153
5FE716F5208B0AB1D63011C7 79F7026C690BAA50B92CD8B66A3AD3F4F22C4FED 9BA44C2621385CB966EBA586F72C284D731FABEE A276772899
86DB50844682F8ACB77FC2E86E29AC A9C5DF4D22E99998D9875A5110C01C5A2F6059E7 DCFD35E0BF8CA7344752DE8B6FB21E8933C60243 F3A0
4C595DB5B6A5F1ECA43E3B7BBB100D811BBE F7DA48BB64BCB84ECBA7EE6935CD23C10D498E23

ENV TOMCAT_MAJOR 8

ENV TOMCAT_VERSION 8.5.32

ENV TOMCAT_SHA512 fc010f4643cb9996cad3812594190564d0a30be717f659110211414faf8063c61fad1f18134154084ad3ddfbbdbb352fa66
86a28fbb6402d3207d4e0a88fa9ce

ENV TOMCAT_TGZ_URLS \

# https://issues.apache.org/jira/browse/INFRA-8753?focusedCommentId=14735394#comment-14735394

https://www.apache.org/dyn/closer.cgi?action=download&filename=tomcat/tomcat-$TOMCAT_MAJOR/v$TOMCAT_VERSION/bin/a

```



```

pache-tomcat-$TOMCAT_VERSION.tar.gz \

# if the version is outdated, we might have to pull from the dist/archive :/

    https://www-us.apache.org/dist/tomcat/tomcat-$TOMCAT_MAJOR/v$TOMCAT_VERSION/bin/apache-tomcat-$TOMCAT_VERSION.tar
.gz \

    https://www.apache.org/dist/tomcat/tomcat-$TOMCAT_MAJOR/v$TOMCAT_VERSION/bin/apache-tomcat-$TOMCAT_VERSION.tar.gz
\

    https://archive.apache.org/dist/tomcat/tomcat-$TOMCAT_MAJOR/v$TOMCAT_VERSION/bin/apache-tomcat-$TOMCAT_VERSION.ta
r.gz

ENV TOMCAT_ASC_URLS \

    https://www.apache.org/dyn/closer.cgi?action=download&filename=tomcat/tomcat-$TOMCAT_MAJOR/v$TOMCAT_VERSION/bin/a
pache-tomcat-$TOMCAT_VERSION.tar.gz.asc \

# not all the mirrors actually carry the .asc files :'(

    https://www-us.apache.org/dist/tomcat/tomcat-$TOMCAT_MAJOR/v$TOMCAT_VERSION/bin/apache-tomcat-$TOMCAT_VERSION.tar
.gz.asc \

    https://www.apache.org/dist/tomcat/tomcat-$TOMCAT_MAJOR/v$TOMCAT_VERSION/bin/apache-tomcat-$TOMCAT_VERSION.tar.gz
.asc \

    https://archive.apache.org/dist/tomcat/tomcat-$TOMCAT_MAJOR/v$TOMCAT_VERSION/bin/apache-tomcat-$TOMCAT_VERSION.ta
r.gz.asc

RUN set -eux; \

\

    savedAptMark="$(apt-mark showmanual)"; \

    apt-get update; \

\

    apt-get install -y --no-install-recommends gnupg dirmngr; \

\

    export GNUPGHOME="$(mktemp -d)"; \

    for key in $GPG_KEYS; do \

        gpg --keyserver ha.pool.sks-keyservers.net --recv-keys "$key"; \

    done; \

\

    apt-get install -y --no-install-recommends wget ca-certificates; \

\

    success=; \

    for url in $TOMCAT_TGZ_URLS; do \

```

```
if wget -O tomcat.tar.gz "$url"; then \  
    success=1; \  
    break; \  
fi; \  
done; \  
[ -n "$success" ]; \  
\  
echo "$TOMCAT_SHA512 *tomcat.tar.gz" | sha512sum -c -; \  
\  
success=; \  
for url in $TOMCAT_ASC_URLS; do \  
    if wget -O tomcat.tar.gz.asc "$url"; then \  
        success=1; \  
        break; \  
    fi; \  
done; \  
[ -n "$success" ]; \  
\  
gpg --batch --verify tomcat.tar.gz.asc tomcat.tar.gz; \  
tar -xvf tomcat.tar.gz --strip-components=1; \  
rm bin/*.bat; \  
rm tomcat.tar.gz*; \  
rm -rf "$GNUPGHOME"; \  
\  
nativeBuildDir="$(mktemp -d)"; \  
tar -xvf bin/tomcat-native.tar.gz -C "$nativeBuildDir" --strip-components=1; \  
apt-get install -y --no-install-recommends \  
    dpkg-dev \  
    gcc \  
    libapr1-dev \  
    libssl-dev \  
    make \  
    "openjdk-${JAVA_VERSION%[.~bu-]*}-jdk=$JAVA_DEBIAN_VERSION" \  

```

```

; \

( \

    export CATALINA_HOME="$PWD"; \

    cd "$nativeBuildDir/native"; \

    gnuArch="$(dpkg-architecture --query DEB_BUILD_GNU_TYPE)"; \

    ./configure \

        --build="$gnuArch" \

        --libdir="$TOMCAT_NATIVE_LIBDIR" \

        --prefix="$CATALINA_HOME" \

        --with-apr="$(which apr-1-config)" \

        --with-java-home="$(docker-java-home)" \

        --with-ssl=yes; \

    make -j "$(nproc)"; \

    make install; \

); \

rm -rf "$nativeBuildDir"; \

rm bin/tomcat-native.tar.gz; \

\

# reset apt-mark's "manual" list so that "purge --auto-remove" will remove all build dependencies

apt-mark auto '.*' > /dev/null; \

[ -z "$savedAptMark" ] || apt-mark manual $savedAptMark; \

apt-get purge -y --auto-remove -o APT::AutoRemove::RecommendsImportant=false; \

rm -rf /var/lib/apt/lists/*; \

\

# sh removes env vars it doesn't support (ones with periods)

# https://github.com/docker-library/tomcat/issues/77

find ./bin/ -name '*.sh' -exec sed -ri 's|^#!/bin/sh$|#!/usr/bin/env bash|' '{}' +

\

# verify Tomcat Native is working properly

RUN set -e \

    && nativeLines="$(catalina.sh configtest 2>&1)" \

    && nativeLines="$(echo "$nativeLines" | grep 'Apache Tomcat Native')"\

```

```
&& nativeLines="$(echo "$nativeLines" | sort -u)" \

&& if ! echo "$nativeLines" | grep 'INFO: Loaded APR based Apache Tomcat Native library' >&2; then \

    echo >&2 "$nativeLines"; \

    exit 1; \

fi
```

EXPOSE 8080

CMD ["catalina.sh", "run"]

通过Dockerfile创建一个镜像，替换成你自己的名字

```
runoob@runoob:~/tomcat$ docker build -t tomcat .
```

创建完成后，我们可以在本地的镜像列表里查找到刚刚创建的镜像

```
runoob@runoob:~/tomcat$ docker images|grep tomcat
```

tomcat	latest	70f819d3d2d9	7 days ago	335.8 MB
--------	--------	--------------	------------	----------

## 使用tomcat镜像

### 运行容器

```
runoob@runoob:~/tomcat$ docker run --name tomcat -p 8080:8080 -v $PWD/test:/usr/local/tomcat/webapps/test -d tomcat

acb33fcb44beb8d7f1ebace6f50f5fc204b1dbe9d524881267aa715c61cf75320

runoob@runoob:~/tomcat$
```

命令说明：

**-p 8080:8080:** 将容器的8080端口映射到主机的8080端口

**-v \$PWD/test:/usr/local/tomcat/webapps/test:** 将主机中当前目录下的test挂载到容器的/test

查看容器启动情况

```
runoob@runoob:~/tomcat$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	... PORTS	NAMES
acb33fcb44beb	tomcat	"catalina.sh run"	... 0.0.0.0:8080->8080/tcp	tomcat

通过浏览器访问

## Apache Tomcat/8.0.35

 The Apache Software Foundation  
<http://www.apache.org/>

If you're seeing this, you've successfully installed Tomcat. Congratulations!



## Recommended Reading:

[Security Considerations HOW-TO](#)[Manager Application HOW-TO](#)[Clustering/Session Replication HOW-TO](#)[Server Status](#)[Manager App](#)[Host Manager](#)

## Developer Quick Start

[□ Docker 安装 PHP](#)[Docker 安装 Python □](#)[□ 点我分享笔记](#)[反馈/建议](#)Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号: 闽ICP备15012807号-1[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)[□ Docker 安装 Tomcat](#)[Docker 安装 Redis □](#)

## Docker 安装 Python

### 方法一、docker pull python:3.5

查找Docker Hub上的python镜像

runoob@runoob:~/python\$ docker search python

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
python	Python is an interpreted,...	982	[OK]	
kaggle/python	Docker image for Python...	33		[OK]
azukiapp/python	Docker image to run Python ...	3		[OK]
vimagick/python	mini python		2	[OK]
tsuru/python	Image for the Python ...	2		[OK]
pandada8/alpine-python	An alpine based python image		1	[OK]
1science/python	Python Docker images based on ...	1		[OK]
lucidfrontier45/python-uwsgi	Python with uWSGI	1		[OK]

orbweb/python	Python image	1	[OK]
pathwar/python	Python template for Pathwar levels 1		[OK]
rounds/10m-python	Python, setuptools and pip.	0	[OK]
ruimashita/python	ubuntu 14.04 python	0	[OK]
tnanba/python	Python on CentOS-7 image.	0	[OK]

这里我们拉取官方的镜像,标签为3.5

```
runoob@runoob:~/python$ docker pull python:3.5
```

等待下载完成后,我们就可以在本地镜像列表里查到REPOSITORY为python,标签为3.5的镜像。

```
runoob@runoob:~/python$ docker images python:3.5
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
python	3.5	045767ddf24a	9 days ago	684.1 MB

## 方法二、通过 Dockerfile 构建

创建Dockerfile

首先, 创建目录python,用于存放后面的相关东西。

```
runoob@runoob:~$ mkdir -p ~/python ~/python/myapp
```

myapp目录将映射为python容器配置的应用目录

进入创建的python目录, 创建Dockerfile

```
FROM buildpack-deps:jessie

# remove several traces of debian python

RUN apt-get purge -y python.*

# http://bugs.python.org/issue19846

# > At the moment, setting "LANG=C" on a Linux system *fundamentally breaks Python 3*, and that's not OK.

ENV LANG C.UTF-8

# gpg: key F73C700D: public key "Larry Hastings <larry@hastings.org>" imported

ENV GPG_KEY 97FC712E4C024BBEA48A61ED3A5CA953F73C700D
```

```
ENV PYTHON_VERSION 3.5.1
```

```
# if this is called "PIP_VERSION", pip explodes with "ValueError: invalid truth value '<VERSION>'"
```

```
ENV PYTHON_PIP_VERSION 8.1.2
```

```
RUN set -ex \
```

```
&& curl -fSL "https://www.python.org/ftp/python/${PYTHON_VERSION%%[a-z]*}/Python-${PYTHON_VERSION}.tar.xz" -o python.tar.xz \
```

```
&& curl -fSL "https://www.python.org/ftp/python/${PYTHON_VERSION%%[a-z]*}/Python-${PYTHON_VERSION}.tar.xz.asc" -o python.tar.xz.asc \
```

```
&& export GNUPGHOME="$(mktemp -d)" \
```

```
&& gpg --keyserver ha.pool.sks-keyservers.net --recv-keys "$GPG_KEY" \
```

```
&& gpg --batch --verify python.tar.xz.asc python.tar.xz \
```

```
&& rm -r "$GNUPGHOME" python.tar.xz.asc \
```

```
&& mkdir -p /usr/src/python \
```

```
&& tar -xJC /usr/src/python --strip-components=1 -f python.tar.xz \
```

```
&& rm python.tar.xz \
```

```
\
```

```
&& cd /usr/src/python \
```

```
&& ./configure --enable-shared --enable-unicode=ucs4 \
```

```
&& make -j$(nproc) \
```

```
&& make install \
```

```
&& ldconfig \
```

```
&& pip3 install --no-cache-dir --upgrade --ignore-installed pip==$PYTHON_PIP_VERSION \
```

```
&& find /usr/local -depth \
```

```
\( \
```

```
\( -type d -a -name test -o -name tests \) \
```

```
-o \
```

```
\( -type f -a -name '*.pyc' -o -name '*.pyo' \) \
```

```
\) -exec rm -rf '{}' + \
```

```
&& rm -rf /usr/src/python ~/.cache
```

```
# make some useful symlinks that are expected to exist
```

```
RUN cd /usr/local/bin \
```

```
&& ln -s easy_install-3.5 easy_install \

&& ln -s idle3 idle \

&& ln -s pydoc3 pydoc \

&& ln -s python3 python \

&& ln -s python3-config python-config
```

CMD ["python3"]

通过Dockerfile创建一个镜像，替换成你自己的名字

```
runoob@runoob:~/python$ docker build -t python:3.5 .
```

创建完成后，我们可以在本地的镜像列表里查找到刚刚创建的镜像

```
runoob@runoob:~/python$ docker images python:3.5
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
python	3.5	045767ddf24a	9 days ago	684.1 MB

## 使用python镜像

在~/python/myapp目录下创建一个 helloworld.py 文件，代码如下：

```
#!/usr/bin/python

print("Hello, World!");
```

## 运行容器

```
runoob@runoob:~/python$ docker run -v $PWD/myapp:/usr/src/myapp -w /usr/src/myapp python:3.5 python helloworld.py
```

命令说明：

**-v \$PWD/myapp:/usr/src/myapp** :将主机中当前目录下的myapp挂载到容器的/usr/src/myapp

**-w /usr/src/myapp** :指定容器的/usr/src/myapp目录为工作目录

**python helloworld.py** :使用容器的python命令来执行工作目录中的helloworld.py文件

输出结果：

```
Hello, World!
```





## Docker 安装 Redis

### 方法一、docker pull redis:3.2

查找Docker Hub上的redis镜像

```
runoob@runoob:~/redis$ docker search redis
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
redis	Redis is an open source ...	2321	[OK]	
sameersbn/redis		32		[OK]
torusware/speedus-redis	Always updated official ...	29		[OK]
bitnami/redis	Bitnami Redis Docker Image	22		[OK]
anapsix/redis	11MB Redis server image ...	6		[OK]
webhippie/redis	Docker images for redis	4		[OK]
clue/redis-benchmark	A minimal docker image t...	3		[OK]
williamyeh/redis	Redis image for Docker	3		[OK]
unblibraries/redis	Leverages phusion/baseim...	2		[OK]
greytip/redis	redis 3.0.3	1		[OK]
servivum/redis	Redis Docker Image	1		[OK]
...				

这里我们拉取官方的镜像,标签为3.2

```
runoob@runoob:~/redis$ docker pull redis:3.2
```

等待下载完成后，我们就可以在本地镜像列表里查到REPOSITORY为redis,标签为3.2的镜像。

```
runoob@runoob:~/redis$ docker images redis
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
redis	3.2	43c923d57784	2 weeks ago	193.9 MB

## 方法二、通过 Dockerfile 构建

创建Dockerfile

首先，创建目录redis,用于存放后面的相关东西。

```
runoob@runoob:~$ mkdir -p ~/redis ~/redis/data
```

data目录将映射为redis容器配置的/data目录,作为redis数据持久化的存储目录

进入创建的redis目录，创建Dockerfile

```
FROM debian:jessie

# add our user and group first to make sure their IDs get assigned consistently, regardless of whatever dependencies
get added

RUN groupadd -r redis && useradd -r -g redis redis

RUN apt-get update && apt-get install -y --no-install-recommends \

    ca-certificates \

    wget \

    && rm -rf /var/lib/apt/lists/*

# grab gosu for easy step-down from root

ENV GOSU_VERSION 1.7

RUN set -x \

    && wget -O /usr/local/bin/gosu "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$(dpkg --print-architecture)" \

    && wget -O /usr/local/bin/gosu.asc "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$(dpkg --print-architecture).asc" \

    && export GNUPGHOME="$(mktemp -d)" \

    && gpg --keyserver ha.pool.sks-keyservers.net --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4 \

    && gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu \

    && rm -r "$GNUPGHOME" /usr/local/bin/gosu.asc \
```

```

    && chmod +x /usr/local/bin/gosu \

    && gosu nobody true

ENV REDIS_VERSION 3.2.0

ENV REDIS_DOWNLOAD_URL http://download.redis.io/releases/redis-3.2.0.tar.gz

ENV REDIS_DOWNLOAD_SHA1 0c1820931094369c8cc19fc1be62f598bc5961ca

# for redis-sentinel see: http://redis.io/topics/sentinel

RUN buildDeps='gcc libc6-dev make' \

    && set -x \

    && apt-get update && apt-get install -y $buildDeps --no-install-recommends \

    && rm -rf /var/lib/apt/lists/* \

    && wget -O redis.tar.gz "$REDIS_DOWNLOAD_URL" \

    && echo "$REDIS_DOWNLOAD_SHA1 *redis.tar.gz" | sha1sum -c - \

    && mkdir -p /usr/src/redis \

    && tar -xzf redis.tar.gz -C /usr/src/redis --strip-components=1 \

    && rm redis.tar.gz \

    && make -C /usr/src/redis \

    && make -C /usr/src/redis install \

    && rm -r /usr/src/redis \

    && apt-get purge -y --auto-remove $buildDeps

RUN mkdir /data && chown redis:redis /data

VOLUME /data

WORKDIR /data

COPY docker-entrypoint.sh /usr/local/bin/

ENTRYPOINT ["docker-entrypoint.sh"]

EXPOSE 6379

CMD [ "redis-server" ]

```

```
runoob@runoob:~/redis$ docker build -t redis:3.2 .
```

创建完成后，我们可以在本地的镜像列表里查找到刚刚创建的镜像

```
runoob@runoob:~/redis$ docker images redis
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
redis	3.2	43c923d57784	2 weeks ago	193.9 MB

## 使用redis镜像

运行容器

```
runoob@runoob:~/redis$ docker run -p 6379:6379 -v $PWD/data:/data -d redis:3.2 redis-server --appendonly yes

43f7a65ec7f8bd64eb1c5d82bc4fb60e5eb31915979c4e7821759aac3b62f330

runoob@runoob:~/redis$
```

命令说明：

**-p 6379:6379** : 将容器的6379端口映射到主机的6379端口

**-v \$PWD/data:/data** : 将主机中当前目录下的data挂载到容器的/data

**redis-server --appendonly yes** : 在容器执行redis-server启动命令，并打开redis持久化配置

查看容器启动情况

```
runoob@runoob:~/redis$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	...	PORTS	NAMES
43f7a65ec7f8	redis:3.2	"docker-entrypoint.sh"	...	0.0.0.0:6379->6379/tcp	agitated_cray

连接、查看容器

使用redis镜像执行redis-cli命令连接到刚启动的容器,主机IP为172.17.0.1

```
runoob@runoob:~/redis$ docker exec -it 43f7a65ec7f8 redis-cli

172.17.0.1:6379> info

# Server

redis_version:3.2.0

redis_git_sha1:00000000

redis_git_dirty:0

redis_build_id:f449541256e7d446

redis_mode:standalone
```

```
os:linux 4.2.0-16-generic x86_64

arch_bits:64

multiplexing_api:epoll

...
```

[❏ Docker 安装 Python](#)

[Docker 安装 MongoDB](#) ❏

[❏ 点我分享笔记](#)

反馈/建议



[❏ Docker 安装 Redis](#)

[Docker 安装 Apache](#) ❏

# Docker 安装 MongoDB

## 方法一、docker pull mongo

查找Docker Hub上的mongo镜像

```
runoob@runoob:~/mongo$ docker search mongo
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
mongo	MongoDB document databases ...	1989	[OK]	
mongo-express	Web-based MongoDB admin int...	22	[OK]	
mvertes/alpine-mongo	light MongoDB container	19		[OK]
mongooseim/mongooseim-docker	MongooseIM server the lates...	9		[OK]
torusware/speedus-mongo	Always updated official Mon...	9		[OK]
jacksoncage/mongo	Instant MongoDB sharded cluster	6		[OK]
mongoclient/mongoclient	Official docker image for M...	4		[OK]
jadsonlourenco/mongo-rocks	Percona Mongodb with Rocksdb...	4		[OK]
asteris/apache-php-mongo	Apache2.4 + PHP + Mongo + m...	2		[OK]
19hz/mongo-container	Mongodb replicaset for coreos	1		[OK]
nitra/mongo	Mongo3 centos7	1		[OK]

ackee/mongo	MongoDB with fixed Bluemix p...	1	[OK]
kobotoolbox/mongo	https://github.com/kobotoolb...	1	[OK]
valtlfelipe/mongo	Docker Image based on the la...	1	[OK]

这里我们拉取官方的镜像,标签为3.2

```
runoob@runoob:~/mongo$ docker pull mongo
```

等待下载完成后,我们就可以在本地镜像列表里查到REPOSITORY为mongo,标签为3.2的镜像。

```
runoob@runoob:~/mongo$ docker images mongo
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mongo	latest	63c6b736e399	2 days ago	379MB

## 方法二、通过 Dockerfile 构建

创建Dockerfile

首先,创建目录mongo,用于存放后面的相关东西。

```
runoob@runoob:~$ mkdir -p ~/mongo ~/mongo/db
```

db目录将映射为mongo容器配置的/data/db目录,作为mongo数据的存储目录

进入创建的mongo目录,创建Dockerfile

```
FROM debian:jessie-slim

# add our user and group first to make sure their IDs get assigned consistently, regardless of whatever dependencies
get added

RUN groupadd -r mongodb && useradd -r -g mongodb mongodb

RUN apt-get update \

    && apt-get install -y --no-install-recommends \

        ca-certificates \

        jq \

        numactl \

    && rm -rf /var/lib/apt/lists/*
```

```

# grab gosu for easy step-down from root (https://github.com/tianon/gosu/releases)

ENV GOSU_VERSION 1.10

# grab "js-yaml" for parsing mongod's YAML config files (https://github.com/nodeca/js-yaml/releases)

ENV JSYAML_VERSION 3.10.0


RUN set -ex; \

\

apt-get update; \

apt-get install -y --no-install-recommends \

    wget \

; \

rm -rf /var/lib/apt/lists/*; \

\

dpkgArch="$(dpkg --print-architecture | awk -F- '{ print $NF }')"; \

wget -O /usr/local/bin/gosu "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch"; \

wget -O /usr/local/bin/gosu.asc "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$dpkgArch.asc"; \

export GNUPGHOME="$(mktemp -d)"; \

gpg --keyserver ha.pool.sks-keyservers.net --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4; \

gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu; \

command -v gpgconf && gpgconf --kill all || :; \

rm -r "$GNUPGHOME" /usr/local/bin/gosu.asc; \

chmod +x /usr/local/bin/gosu; \

gosu nobody true; \

\

wget -O /js-yaml.js "https://github.com/nodeca/js-yaml/raw/${JSYAML_VERSION}/dist/js-yaml.js"; \

# TODO some sort of download verification here

\

apt-get purge -y --auto-remove wget


RUN mkdir /docker-entrypoint-initdb.d


ENV GPG_KEYS \

```

```

# pub 4096R/AAB2461C 2014-02-25 [expires: 2016-02-25]

# Key fingerprint = DFFA 3DCF 326E 302C 4787 673A 01C4 E7FA AAB2 461C

# uid MongoDB 2.6 Release Signing Key <packaging@mongodb.com>

DFFA3DCF326E302C4787673A01C4E7FAAAB2461C \

# pub 4096R/EA312927 2015-10-09 [expires: 2017-10-08]

# Key fingerprint = 42F3 E95A 2C4F 0827 9C49 60AD D68F A50F EA31 2927

# uid MongoDB 3.2 Release Signing Key <packaging@mongodb.com>

42F3E95A2C4F08279C4960ADD68FA50FEA312927

# https://docs.mongodb.com/manual/tutorial/verify-mongodb-packages/#download-then-import-the-key-file

RUN set -ex; \

    export GNUPGHOME="$(mktemp -d)"; \

    for key in $GPG_KEYS; do \

        gpg --keyserver ha.pool.sks-keyservers.net --recv-keys "$key"; \

    done; \

    gpg --export $GPG_KEYS > /etc/apt/trusted.gpg.d/mongodb.gpg; \

    command -v gpgconf && gpgconf --kill all || :; \

    rm -r "$GNUPGHOME"; \

    apt-key list


# Allow build-time overrides (eg. to build image with MongoDB Enterprise version)

# Options for MONGO_PACKAGE: mongodb-org OR mongodb-enterprise

# Options for MONGO_REPO: repo.mongodb.org OR repo.mongodb.com

# Example: docker build --build-arg MONGO_PACKAGE=mongodb-enterprise --build-arg MONGO_REPO=repo.mongodb.com .

ARG MONGO_PACKAGE=mongodb-org

ARG MONGO_REPO=repo.mongodb.org

ENV MONGO_PACKAGE=${MONGO_PACKAGE} MONGO_REPO=${MONGO_REPO}


ENV MONGO_MAJOR 3.2

ENV MONGO_VERSION 3.2.20


RUN echo "deb http://$MONGO_REPO/apt/debian jessie/${MONGO_PACKAGE%-unstable}/${MONGO_MAJOR} main" | tee "/etc/apt/sources.list.d/${MONGO_PACKAGE%-unstable}.list"

```



```

RUN set -x \

    && apt-get update \

    && apt-get install -y \

        ${MONGO_PACKAGE}=${MONGO_VERSION} \

        ${MONGO_PACKAGE}-server=${MONGO_VERSION} \

        ${MONGO_PACKAGE}-shell=${MONGO_VERSION} \

        ${MONGO_PACKAGE}-mongos=${MONGO_VERSION} \

        ${MONGO_PACKAGE}-tools=${MONGO_VERSION} \

    && rm -rf /var/lib/apt/lists/* \

    && rm -rf /var/lib/mongodb \

    && mv /etc/mongod.conf /etc/mongod.conf.orig

RUN mkdir -p /data/db /data/configdb \

    && chown -R mongodb:mongodb /data/db /data/configdb

VOLUME /data/db /data/configdb

COPY docker-entrypoint.sh /usr/local/bin/

RUN ln -s usr/local/bin/docker-entrypoint.sh /entrypoint.sh # backwards compat

ENTRYPOINT ["docker-entrypoint.sh"]

EXPOSE 27017

CMD ["mongod"]

```

通过Dockerfile创建一个镜像，替换成你自己的名字

```
runoob@runoob:~/mongo$ docker build -t mongo:3.2 .
```

创建完成后，我们可以在本地的镜像列表里查找到刚刚创建的镜像

```
runoob@runoob:~/mongo$ docker images mongo:3.2
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mongo	3.2	282fd552add6	9 days ago	336.1 MB

# 使用mongo镜像

运行容器

```
runoob@runoob:~/mongo$ docker run -p 27017:27017 -v $PWD/db:/data/db -d mongo:3.2

cda8830cad5fe35e9c4aed037bbd5434b69b19bf2075c8626911e6ebb08cad51

runoob@runoob:~/mongo$
```

命令说明:

**-p 27017:27017** :将容器的27017 端口映射到主机的27017 端口

**-v \$PWD/db:/data/db** :将主机中当前目录下的db挂载到容器的/data/db, 作为mongo数据存储目录

查看容器启动情况

```
runoob@runoob:~/mongo$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	...	PORTS	NAMES
cda8830cad5f	mongo:3.2	"/entrypoint.sh mongo"	...	0.0.0.0:27017->27017/tcp	suspicious_goodall

使用mongo镜像执行mongo 命令连接到刚启动的容器,主机IP为172.17.0.1

```
runoob@runoob:~/mongo$ docker run -it mongo:3.2 mongo --host 172.17.0.1

MongoDB shell version: 3.2.7

connecting to: 172.17.0.1:27017/test

Welcome to the MongoDB shell.

For interactive help, type "help".

For more comprehensive documentation, see

  http://docs.mongodb.org/

Questions? Try the support group

  http://groups.google.com/group/mongodb-user

>
```

[□ Docker 安装 Redis](#)

[Docker 安装 Apache □](#)

[□ 点我分享笔记](#)

反馈/建议



# Docker 安装 Apache

## 方法一、docker pull httpd

查找Docker Hub上的httpd镜像

```
runoob@runoob:~/apache$ docker search httpd
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
httpd	The Apache HTTP Server ..	524	[OK]	
centos/httpd		7		[OK]
rgielen/httpd-image-php5	Docker image for Apache...	1		[OK]
microwebapps/httpd-frontend	Httpd frontend allowing...	1		[OK]
lolhens/httpd	Apache httpd 2 Server	1		[OK]
publici/httpd	httpd:latest	0		[OK]
publicisworldwide/httpd	The Apache httpd webser...	0		[OK]
rgielen/httpd-image-simple	Docker image for simple...	0		[OK]
solsson/httpd	Derivatives of the offi...	0		[OK]
rgielen/httpd-image-drush	Apache HTTPD + Drupal S...	0		[OK]
learninglayers/httpd		0		[OK]
sohrabkhan/httpd	Docker httpd + php5.6 (...)	0		[OK]
aintohvri/docker-httpd	Apache HTTPD Docker ext...	0		[OK]
alizarion/httpd	httpd on centos with mo...	0		[OK]
...				

这里我们拉取官方的镜像

```
runoob@runoob:~/apache$ docker pull httpd
```

等待下载完成后，我们就可以在本地镜像列表里查到REPOSITORY为httpd的镜像。

```
runoob@runoob:~/apache$ docker images httpd
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

httpd latest da1536b4ef14 23 seconds ago 195.1 MB

## 方法二、通过 Dockerfile 构建

### 创建 Dockerfile

首先，创建目录 `apache`，用于存放后面的相关东西。

```
runoob@runoob:~$ mkdir -p ~/apache/www ~/apache/logs ~/apache/conf
```

`www` 目录将映射为 `apache` 容器配置的应用程序目录

`logs` 目录将映射为 `apache` 容器的日志目录

`conf` 目录里的配置文件将映射为 `apache` 容器的配置文件

进入创建的 `apache` 目录，创建 `Dockerfile`

```
FROM debian:jessie

# add our user and group first to make sure their IDs get assigned consistently, regardless of whatever dependencies
get added

#RUN groupadd -r www-data && useradd -r --create-home -g www-data www-data

ENV HTTPD_PREFIX /usr/local/apache2

ENV PATH $PATH:$HTTPD_PREFIX/bin

RUN mkdir -p "$HTTPD_PREFIX" \

    && chown www-data:www-data "$HTTPD_PREFIX"

WORKDIR $HTTPD_PREFIX

# install httpd runtime dependencies

# https://httpd.apache.org/docs/2.4/install.html#requirements

RUN apt-get update \

    && apt-get install -y --no-install-recommends \

    libapr1 \

    libaprutil1 \

    libaprutil1-ldap \

    libapr1-dev \

    libaprutil1-dev \

    libpcre++0 \

    libssl1.0.0 \
```

```
&& rm -r /var/lib/apt/lists/*
```

```
ENV HTTPD_VERSION 2.4.20
```

```
ENV HTTPD_BZ2_URL https://www.apache.org/dist/httpd/httpd-$HTTPD_VERSION.tar.bz2
```

```
RUN buildDeps=' \
```

```
    ca-certificates \
```

```
    curl \
```

```
    bzip2 \
```

```
    gcc \
```

```
    libpcre++-dev \
```

```
    libssl-dev \
```

```
    make \
```

```
' \
```

```
set -x \
```

```
&& apt-get update \
```

```
&& apt-get install -y --no-install-recommends $buildDeps \
```

```
&& rm -r /var/lib/apt/lists/* \
```

```
\
```

```
&& curl -fSL "$HTTPD_BZ2_URL" -o httpd.tar.bz2 \
```

```
&& curl -fSL "$HTTPD_BZ2_URL.asc" -o httpd.tar.bz2.asc \
```

```
# see https://httpd.apache.org/download.cgi#verify
```

```
&& export GNUPGHOME="$(mktemp -d)" \
```

```
&& gpg --keyserver ha.pool.sks-keyservers.net --recv-keys A93D62ECC3C8EA12DB220EC934EA76E6791485A8 \
```

```
&& gpg --batch --verify httpd.tar.bz2.asc httpd.tar.bz2 \
```

```
&& rm -r "$GNUPGHOME" httpd.tar.bz2.asc \
```

```
\
```

```
&& mkdir -p src \
```

```
&& tar -xvf httpd.tar.bz2 -C src --strip-components=1 \
```

```
&& rm httpd.tar.bz2 \
```

```
&& cd src \
```

```
\
```

```

&& ./configure \

    --prefix="$HTTPD_PREFIX" \

    --enable-mods-shared=reallyall \

&& make -j"${nproc}" \

&& make install \

\

&& cd .. \

&& rm -r src \

\

&& sed -ri \

    -e 's!^(\\s*CustomLog)\\s+\\S+!\\1 /proc/self/fd/1!g' \

    -e 's!^(\\s*ErrorLog)\\s+\\S+!\\1 /proc/self/fd/2!g' \

    "$HTTPD_PREFIX/conf/httpd.conf" \

\

&& apt-get purge -y --auto-remove $buildDeps

COPY httpd-foreground /usr/local/bin/

EXPOSE 80

CMD ["httpd-foreground"]

```

Dockerfile文件中 `COPY httpd-foreground /usr/local/bin/` 是将当前目录下的`httpd-foreground`拷贝到镜像里，作为`httpd`服务的启动脚本，所以我们要在本地创建一个脚本文件`httpd-foreground`

```

#!/bin/bash

set -e

# Apache gets grumpy about PID files pre-existing

rm -f /usr/local/apache2/logs/httpd.pid

exec httpd -DFOREGROUND

```

赋予`httpd-foreground`文件可执行权限

```
runoob@runoob:~/apache$ chmod +x httpd-foreground
```

通过Dockerfile创建一个镜像，替换成你自己的名字

```
runoob@runoob:~/apache$ docker build -t httpd .
```

创建完成后，我们可以在本地的镜像列表里查找到刚刚创建的镜像

```
runoob@runoob:~/apache$ docker images httpd
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	latest	da1536b4ef14	23 seconds ago	195.1 MB

## 使用apache镜像

### 运行容器

```
docker run -p 80:80 -v $PWD/www:/usr/local/apache2/htdocs/ -v $PWD/conf/httpd.conf:/usr/local/apache2/conf/httpd.conf -v $PWD/logs:/usr/local/apache2/logs/ -d httpd
```

命令说明：

**-p 80:80** :将容器的80端口映射到主机的80端口

**-v \$PWD/www:/usr/local/apache2/htdocs/** :将主机中当前目录下的www目录挂载到容器的/usr/local/apache2/htdocs/

**-v \$PWD/conf/httpd.conf:/usr/local/apache2/conf/httpd.conf** :将主机中当前目录下的conf/httpd.conf文件挂载到容器的/usr/local/apache2/conf/httpd.conf

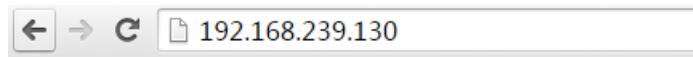
**-v \$PWD/logs:/usr/local/apache2/logs/** :将主机中当前目录下的logs目录挂载到容器的/usr/local/apache2/logs/

查看容器启动情况

```
runoob@runoob:~/apache$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	... PORTS	NAMES
79a97f2aac37	httpd	"httpd-foreground"	... 0.0.0.0:80->80/tcp	sharp_swanson

通过浏览器访问



It works! <http://www.runoob.com/>

[❏ Docker 安装 MongoDB](#)

[Docker 命令大全](#) ❏

[❏ 点我分享笔记](#)

[反馈/建议](#)



# Docker 命令大全

## 容器生命周期管理

- run
- start/stop/restart
- kill
- rm
- pause/unpause
- create
- exec

## 容器操作

- ps
- inspect
- top
- attach
- events
- logs
- wait
- export
- port

## 容器rootfs命令

- commit
- cp
- diff

## 镜像仓库

- login
- pull
- push
- search

## 本地镜像管理

- images
- rmi
- tag
- build



history

save

import

## info|version

info

version

☐ Docker 安装 Apache

Docker run 命令 ☐

☐ 点我分享笔记

反馈/建议

Copyright © 2013-2018 菜鸟教程 [runoob.com](http://runoob.com) All Rights Reserved. 备案号：闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

☐ Docker version 命令

MacOS Docker 安装 ☐

## Docker 资源汇总

### Docker官方英文资源

docker官网: <http://www.docker.com>

Docker windows入门: <https://docs.docker.com/windows/>

Docker Linux 入门: <https://docs.docker.com/linux/>

Docker mac 入门: <https://docs.docker.com/mac/>

Docker 用户指引: <https://docs.docker.com/engine/userguide/>

Docker 官方博客: <http://blog.docker.com/>

Docker Hub: <https://hub.docker.com/>

Docker开源: <https://www.docker.com/open-source>

### Docker中文资源

Docker中文网站: <https://www.docker-cn.com/>

Docker安装手册: <https://docs.docker-cn.com/engine/installation/>

### Docker 国内镜像

网易加速器: <http://hub-mirror.c.163.com>

官方中国加速器: <https://registry.docker-cn.com>

ustc的镜像: <https://docker.mirrors.ustc.edu.cn>

daocloud: <https://www.daocloud.io/mirror#accelerator-doc> (注册后使用)

如果有更好的资源, 欢迎通过右下角的反馈按钮发邮件给我们.....

☐ Docker version 命令

MacOS Docker 安装 ☐

[□ 点我分享笔记](#)

[反馈/建议](#)