Docker官方文档 时间：16年11月5日

# 介绍

Docker is an open-source engine to easily create lightweight, portable, self-sufficient containers from any application. （docker是一个开源引擎，它能够很容易的为任何引用创建轻量级、便携的自给自足的容器。）The same container that a developer builds and tests on a laptop can run at scale, in production, on VMs, bare metal, OpenStack clusters, or any major infrastructure provider.（开发者在笔记本上建立和测试的这个容器能够运行在任何规模的生产环境中，如在VM上，电脑裸机，openstack集群或者任何主要的基础架构上）

Common use cases for Docker include: （docker通常应用在一下方面）

* Automating the packaging and deployment of web applications.

使得web应用的包管理和开发自动化

* Automated testing and continuous integration/deployment.

使得测试和开发整合的持续性自动化

* Deploying and scaling databases and backend services in a service-oriented environment.

在一个以服务为导向的环境中部署和洁制数据库和基础搭建服务

* Building custom PaaS environments, either from scratch or as an extension of off-the-shelf platforms like OpenShift or Cloud Foundry.

建立一个定制化的PaaS环境，或者作为像openshift或者基于云的扩展。

请注意，Docker目前正在沉重的研究与开发。它不应该在生产中使用（还）。

For a high-level overview of Docker, please see the [Introduction](http://www.docker.io/learn_more/). When you’re ready to start working with Docker, we have a [quick start](http://www.docker.io/gettingstarted) and a more in-depth guide to*[Ubuntu](http://docker-doc.readthedocs.io/zh_CN/latest/installation/ubuntulinux.html" \l "ubuntu-linux)* and other *[安装](http://docker-doc.readthedocs.io/zh_CN/latest/installation/index.html" \l "installation-list)* paths including prebuilt binaries, Vagrant-created VMs, Rackspace and Amazon instances.

如果想深入了解DOCKER,请参考[Introduction](http://www.docker.io/learn_more/).。当你已经准备好开始伴随DOCKER工作，我们有一个快速开始[quick start](http://www.docker.io/gettingstarted) 和针对ubuntu和其他安装方式如预建二进制、Vagrant-created VMs以及亚马逊等例子的一个更加深入的指南。

### 安装

有很多系统版本的安装指南，这里只翻译docker在ubuntu 和 red hat中的安装。

# **Ubuntu--ubuntu下安装**

警告

由于下载太慢，需要修改hosts文件,或者修改docker.conf文件

vim /etc/hosts

54.234.135.251 get.docker.io

54.234.135.251 cdn-registry-1.docker.io

vim /etc/init/docker.conf*#在respawn后加*

evn HTTP\_PROXY="http://192.241.209.203:8384"

警告

These instructions have changed for 0.6. If you are upgrading from an earlier version, you will need to follow them again.

注解

Docker is still under heavy development! We don’t recommend using it in production yet, but we’re getting closer with each release. Please see our blog post, [“Getting to Docker 1.0”](http://blog.docker.io/2013/08/getting-to-docker-1-0/)

Docker is supported on the following versions of Ubuntu:

* *[Ubuntu Precise 12.04 (LTS) (64-bit)](http://docker-doc.readthedocs.io/zh_CN/latest/installation/ubuntulinux.html" \l "ubuntu-precise)*
* *[Ubuntu Raring 13.04 and Saucy 13.10 (64 bit)](http://docker-doc.readthedocs.io/zh_CN/latest/installation/ubuntulinux.html" \l "ubuntu-raring-saucy)*

Please read *[Docker和防火墙](http://docker-doc.readthedocs.io/zh_CN/latest/installation/ubuntulinux.html" \l "ufw)*, if you plan to use [UFW (Uncomplicated Firewall)](https://help.ubuntu.com/community/UFW)

## **Ubuntu Precise 12.04 (LTS) (64-bit)**

This installation path should work at all times.

### **依赖关系**

****Linux kernel 3.8****

Due to a bug in LXC, Docker works best on the 3.8 kernel. Precise comes with a 3.2 kernel, so we need to upgrade it. The kernel you’ll install when following these steps comes with AUFS built in. We also include the generic headers to enable packages that depend on them, like ZFS and the VirtualBox guest additions. If you didn’t install the headers for your “precise” kernel, then you can skip these headers for the “raring” kernel. But it is safer to include them if you’re not sure.

*# install the backported kernel*

sudo apt-get update

sudo apt-get install linux-image-generic-lts-raring linux-headers-generic-lts-raring

*# reboot*

sudo reboot

### **安装**

警告

These instructions have changed for 0.6. If you are upgrading from an earlier version, you will need to follow them again.

Docker is available as a Debian package, which makes installation easy. ****See the :ref:`installmirrors` section below if you are not in the United States.**** Other sources of the Debian packages may be faster for you to install.

First add the Docker repository key to your local keychain.

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9

Add the Docker repository to your apt sources list, update and install the lxc-docker package.

*You may receive a warning that the package isn’t trusted. Answer yes to continue installation.*

sudo sh -c "echo deb http://get.docker.io/ubuntu docker main\> /etc/apt/sources.list.d/docker.list"

sudo apt-get update

sudo apt-get install lxc-docker

注解

There is also a simple curl script available to help with this process.

curl -s https://get.docker.io/ubuntu/ | sudo sh

Now verify that the installation has worked by downloading the ubuntu image and launching a container.

sudo docker run -i -t ubuntu /bin/bash

Type exit to exit

****Done!****, now continue with the *[Hello World](http://docker-doc.readthedocs.io/zh_CN/latest/examples/hello_world.html" \l "id1)* example.

## **Ubuntu Raring 13.04 and Saucy 13.10 (64 bit)**

These instructions cover both Ubuntu Raring 13.04 and Saucy 13.10.

### **依赖关系**

****Optional AUFS filesystem support****

Ubuntu Raring already comes with the 3.8 kernel, so we don’t need to install it. However, not all systems have AUFS filesystem support enabled. AUFS support is optional as of version 0.7, but it’s still available as a driver and we recommend using it if you can.

To make sure AUFS is installed, run the following commands:

sudo apt-get update

sudo apt-get install linux-image-extra-`uname -r`

### **安装**

Docker is available as a Debian package, which makes installation easy.

警告

Please note that these instructions have changed for 0.6. If you are upgrading from an earlier version, you will need to follow them again.

First add the Docker repository key to your local keychain.

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9

Add the Docker repository to your apt sources list, update and install the lxc-docker package.

sudo sh -c "echo deb http://get.docker.io/ubuntu docker main\> /etc/apt/sources.list.d/docker.list"

sudo apt-get update

sudo apt-get install lxc-docker

Now verify that the installation has worked by downloading the ubuntu image and launching a container.

sudo docker run -i -t ubuntu /bin/bash

Type exit to exit

****Done!****, now continue with the *[Hello World](http://docker-doc.readthedocs.io/zh_CN/latest/examples/hello_world.html" \l "id1)* example.

### **赋予非root访问权限**

The docker daemon always runs as the root user, and since Docker version 0.5.2, the docker daemon binds to a Unix socket instead of a TCP port. By default that Unix socket is owned by the user *root*, and so, by default, you can access it with sudo.

Starting in version 0.5.3, if you (or your Docker installer) create a Unix group called *docker* and add users to it, then the docker daemon will make the ownership of the Unix socket read/writable by the *docker*group when the daemon starts. The docker daemon must always run as the root user, but if you run thedocker client as a user in the *docker* group then you don’t need to add sudo to all the client commands.

警告

The *docker* group is root-equivalent.

****Example:****

*# Add the docker group if it doesn't already exist.*

sudo groupadd docker

*# Add the connected user "${USER}" to the docker group.# Change the user name to match your preferred user.# You may have to logout and log back in again for# this to take effect.*

sudo gpasswd -a *${*USER*}* docker

*# Restart the Docker daemon.*

sudo service docker restart

### **更新**

To install the latest version of docker, use the standard apt-get method:

*# update your sources list*

sudo apt-get update

*# install the latest*

sudo apt-get install lxc-docker

## **Troubleshooting**

On Linux Mint, the cgroups-lite package is not installed by default. Before Docker will work correctly, you will need to install this via:

sudo apt-get update && sudo apt-get install cgroups-lite

## **Docker和防火墙**

Docker uses a bridge to manage container networking. By default, UFW drops all forwarding traffic. As a result you will need to enable UFW forwarding:

sudo nano /etc/default/ufw

----*# Change:# DEFAULT\_FORWARD\_POLICY="DROP"# to*DEFAULT\_FORWARD\_POLICY="ACCEPT"

Then reload UFW:

sudo ufw reload

UFW’s default set of rules denies all incoming traffic. If you want to be able to reach your containers from another host then you should allow incoming connections on the Docker port (default 4243):

sudo ufw allow 4243/tcp

## **Mirrors**

You should ping get.docker.io and compare the latency to the following mirrors, and pick whichever one is best for you.

### **Yandex**

[Yandex](http://yandex.ru/) in Russia is mirroring the Docker Debian packages, updating every 6 hours. Substitutehttp://mirror.yandex.ru/mirrors/docker/ for http://get.docker.io/ubuntu in the instructions above. For example:

sudo sh -c "echo deb http://mirror.yandex.ru/mirrors/docker/ docker main\> /etc/apt/sources.list.d/docker.list"

sudo apt-get update

sudo apt-get install lxc-docker

# Red Hat Enterprise Linux（red hat企业版linux系统）

注解

Docker is still under heavy development! （目前docker仍然处在繁忙的开发中）We don’t recommend using it in production yet, but we’re getting closer with each release. Please see our blog post, [“Getting to Docker 1.0”](http://blog.docker.io/2013/08/getting-to-docker-1-0/)

（我们不建议将docker用在生产环境中，但是随着每一次版本的更新，我们越来越接近适合生产环境了，请看我们的博客：[“Getting to Docker 1.0”](http://blog.docker.io/2013/08/getting-to-docker-1-0/)）

注解

This is a community contributed installation path. （有一个社区贡献的安装路径）The only ‘official’ installation is using the *[Ubuntu](http://docker-doc.readthedocs.io/zh_CN/latest/installation/ubuntulinux.html" \l "ubuntu-linux)* installation path. （唯一的官方安装使用的是ubuntu）This version may be out of date because it depends on some binaries to be updated and published.

(这一版本可能已经过时了，因为它依赖于一些二进制文件被更新和发布）

Docker is available for ****RHEL**** on EPEL. These instructions should work for both RHEL and CentOS. They will likely work for other binary compatible EL6 distributions as well, but they haven’t been tested.

Red hat 企业linux可以在EPEL上获得docker。这里的说明能够适用于red hat企业linux和centos。这两个系统也可能使得其他二进制编译文件EL6版本工作正常，但是还没有过测试。

Please note that this package is part of [Extra Packages for Enterprise Linux (EPEL)](https://fedoraproject.org/wiki/EPEL), a community effort to create and maintain additional packages for the RHEL distribution.

请注意：这个包是企业linux扩展包组织（[Extra Packages for Enterprise Linux (EPEL)](https://fedoraproject.org/wiki/EPEL)）（一个致力于为red hat 企业版linux提供和创建额外的包的社区）的部分，

Also note that due to the current Docker limitations, Docker is able to run only on the ****64 bit**** architecture.

同时也要注意到，鉴于当前docker的限制，docker只能在64为系统架构上运行。

## **Installation**

Firstly, you need to install the EPEL repository. Please follow the [EPEL installation instructions](https://fedoraproject.org/wiki/EPEL" \l "How_can_I_use_these_extra_packages.3F).

首先，你需要安装EPEL仓库，请参考：[EPEL installation instructions](https://fedoraproject.org/wiki/EPEL" \l "How_can_I_use_these_extra_packages.3F).

The docker-io package provides Docker on EPEL.

Docker-io包在EPEL上提供docker

If you already have the (unrelated) docker package installed, it will conflict with docker-io. There’s a [bug report](https://bugzilla.redhat.com/show_bug.cgi?id=1043676) filed for it. To proceed with docker-io installation, please remove docker first.

如果你已经安装了docker包，它将会与docker-io冲突。这里有一份处理这个问题的bug报告。为了运行安装docker-io，请先移除doceker。

Next, let’s install the docker-io package which will install Docker on our host.

接下来，让我们安装docker-io包，这个将会在我们的主机上安装docker

sudo yum -y install docker-io

To update the docker-io package 更新docker-io包

sudo yum -y update docker-io

Now that it’s installed, let’s start the Docker daemon. 现在docker-io包已经安装好，让我们开启一个docker的实例

sudo service docker start

If we want Docker to start at boot, we should also: 如果我们想让docker作为开机启动项，我们应该执行下面的命令：

sudo chkconfig docker on

Now let’s verify that Docker is working. 现在，让我们确定docker是否工作正常

sudo docker run -i -t fedora /bin/bash

****Done!****, now continue with the *[Hello World](http://docker-doc.readthedocs.io/zh_CN/latest/examples/hello_world.html" \l "id1)* example. 完成，继续执行例子：*[Hello World](http://docker-doc.readthedocs.io/zh_CN/latest/examples/hello_world.html" \l "id1)*

使用Docker第一步

**检测Docker安装**

This guide assumes you have a working installation of Docker.（这个指南假设你已经运行安装了一个docker） To check your Docker install, run the following command:

可以通过执行下面的命令来检测docker是否安装。

# Check that you have a working install 检测运行安装

docker info

If you get docker: command not found or something like /var/lib/docker/repositories: permission denied you may have an incomplete docker installation or insufficient privileges to access Docker on your machine.

Please refer to [安装](http://docker-doc.readthedocs.io/zh_CN/latest/installation/index.html" \l "installation-list) for installation instructions. 请参考安装里面的指南

下载预建的镜像

# Download an ubuntu image --下载一个ubuntu 图像

sudo docker pull ubuntu

This will find the ubuntu image by name in the [Central Index](http://docker-doc.readthedocs.io/zh_CN/latest/use/workingwithrepository.html" \l "searching-central-index) and download it from the top-level Central Repository to a local image cache.

这条命令将会在中心索引中通过名字来寻找一个ubuntu图像，并从顶级中心库下载到本地图像缓存。

注解

When the image has successfully downloaded, you will see a 12 character hash 539c0211cd76: Download complete which is the short form of the image ID. These short image IDs are the first 12 characters of the full image ID - which can be found using docker inspect or docker images -notrunc=true

当这个图片被成功下载下来，你将会看到一个12字节的hash码如539c0211cd76：下载完成，这是图片id的简短形式，这些简短的的图像id是全部图像id的前12个字符。可以通过docker inspect或者docker inages -notrunc=true来查看。

运行一个交互的shell

# Run an interactive shell in the ubuntu image,# allocate a tty, attach stdin and stdout# To detach the tty without exiting the shell,# use the escape sequence Ctrl-p + Ctrl-q

在ubuntu图像中运行一个交互的shell。分配一个tty，附加标准输出和输入，为了离开这个tty而不提出shell，使用逃离序列：ctrl+p 和ctrl+q

sudo docker run -i -t ubuntu /bin/bash

绑定Docker到另一台主机/端口或Unix套接字

警告

Changing the default docker daemon binding to a TCP port or Unix docker user group will increase your security risks by allowing non-root users to potentially gainroot access on the host ([e.g. #1369](https://github.com/dotcloud/docker/issues/1369)). Make sure you control access to docker.

With -H it is possible to make the Docker daemon to listen on a specific IP and port. By default, it will listen on unix:///var/run/docker.sock to allow only local connections by the root user. You could set it to 0.0.0.0:4243 or a specific host IP to give access to everybody, but that is not recommended because then it is trivial for someone to gain root access to the host where the daemon is running.

Similarly, the Docker client can use -H to connect to a custom port.

-H accepts host and port assignment in the following format: tcp://[host][:port] or unix://path

For example:

tcp://host:4243 -> tcp connection on host:4243

unix://path/to/socket -> unix socket located at path/to/socket

-H, when empty, will default to the same value as when no -H was passed in.

-H also accepts short form for TCP bindings: host[:port] or :port

# Run docker in daemon mode

sudo <path to>/docker -H 0.0.0.0:5555 -d &# Download an ubuntu image

sudo docker -H :5555 pull ubuntu

You can use multiple -H, for example, if you want to listen on both TCP and a Unix socket

# Run docker in daemon mode

sudo <path to>/docker -H tcp://127.0.0.1:4243 -H unix:///var/run/docker.sock -d &# Download an ubuntu image, use default Unix socket

sudo docker pull ubuntu# OR use the TCP port

sudo docker -H tcp://127.0.0.1:4243 pull ubuntu

启动长期运行工作进程

# Start a very useful long-running processJOB=$(sudo docker run -d ubuntu /bin/sh -c "while true; do echo Hello world; sleep 1; done")

# Collect the output of the job so far

sudo docker logs $JOB

# Kill the job

sudo docker kill $JOB

列出所有正在运行的容器

sudo docker ps

Bind a service on a TCP port

# Bind port 4444 of this container, and tell netcat to listen on itJOB=$(sudo docker run -d -p 4444 ubuntu:12.10 /bin/nc -l 4444)

# Which public port is NATed to my container?PORT=$(sudo docker port $JOB 4444 | awk -F: '{ print $2 }')

# Connect to the public portecho hello world | nc 127.0.0.1 $PORT

# Verify that the network connection workedecho "Daemon received: $(sudo docker logs $JOB)"

提交（保存）容器状态

Save your containers state to a container image, so the state can be re-used.

When you commit your container only the differences between the image the container was created from and the current state of the container will be stored (as a diff). See which images you already have using the docker images command.

# Commit your container to a new named image

sudo docker commit <container\_id> <some\_name>

# List your containers

sudo docker images

You now have a image state from which you can create new instances.

Read more about [通过存储库分享镜像](http://docker-doc.readthedocs.io/zh_CN/latest/use/workingwithrepository.html" \l "working-with-the-repository) or continue to the complete [Command Line Help](http://docker-doc.readthedocs.io/zh_CN/latest/reference/commandline/cli.html" \l "cli)

# **通过存储库分享镜像**

A *repository* is a hosted collection of tagged *[images](http://docker-doc.readthedocs.io/zh_CN/latest/terms/image.html" \l "image-def)* that together create the file system for a container. The repository’s name is a tag that indicates the provenance of the repository, i.e. who created it and where the original copy is located.

You can find one or more repositories hosted on a *registry*. There can be an implicit or explicit host name as part of the repository tag. The implicit registry is located atindex.docker.io, the home of “top-level” repositories and the Central Index. This registry may also include public “user” repositories.

So Docker is not only a tool for creating and managing your own *[containers](http://docker-doc.readthedocs.io/zh_CN/latest/terms/container.html" \l "container-def)* – ****Docker is also a tool for sharing****. The Docker project provides a Central Registry to host public repositories, namespaced by user, and a Central Index which provides user authentication and search over all the public repositories. You can host your own Registry too! Docker acts as a client for these services via docker search, pull, login and push.

## **公共库**

There are two types of public repositories: *top-level* repositories which are controlled by the Docker team, and *user* repositories created by individual contributors. Anyone can read from these repositories – they really help people get started quickly! You could also use *[可信构建](http://docker-doc.readthedocs.io/zh_CN/latest/use/workingwithrepository.html" \l "using-private-repositories)* if you need to keep control of who accesses your images, but we will only refer to public repositories in these examples.

* Top-level repositories can easily be recognized by ****not**** having a / (slash) in their name. These repositories can generally be trusted.
* User repositories always come in the form of <username>/<repo\_name>. This is what your published images will look like if you push to the public Central Registry.
* Only the authenticated user can push to their *username* namespace on the Central Registry.
* User images are not checked, it is therefore up to you whether or not you trust the creator of this image.

## **在中央索引库查找公共镜像**

You can search the Central Index [online](https://index.docker.io/) or by the CLI. Searching can find images by name, user name or description:

$ sudo docker help search

Usage: docker search NAME

Search the docker index for images

-notrunc=false: Don't truncate output

$ sudo docker search centos

Found 25 results matching your query ("centos")

NAME DESCRIPTION

centos

slantview/centos-chef-solo CentOS 6.4 with chef-solo.

...

There you can see two example results: centos and slantview/centos-chef-solo. The second result shows that it comes from the public repository of a user, slantview/, while the first result (centos) doesn’t explicitly list a repository so it comes from the trusted Central Repository. The / character separates a user’s repository and the image name.

Once you have found the image name, you can download it:

*# sudo docker pull <value>*$ sudo docker pull centos

Pulling repository centos

539c0211cd76: Download complete

What can you do with that image? Check out the *[实例](http://docker-doc.readthedocs.io/zh_CN/latest/examples/index.html" \l "example-list)* and, when you’re ready with your own image, come back here to learn how to share it.

## **贡献给中央档案**

Anyone can pull public images from the Central Registry, but if you would like to share one of your own images, then you must register a unique user name first. You can create your username and login on the [central Docker Index online](https://index.docker.io/account/signup/), or by running

sudo docker login

This will prompt you for a username, which will become a public namespace for your public repositories.

If your username is available then docker will also prompt you to enter a password and your e-mail address. It will then automatically log you in. Now you’re ready to commit and push your own images!

## **提交一个容器到已命名的镜像**

When you make changes to an existing image, those changes get saved to a container’s file system. You can then promote that container to become an image by making a commit. In addition to converting the container to an image, this is also your opportunity to name the image, specifically a name that includes your user name from the Central Docker Index (as you did a login above) and a meaningful name for the image.

*# format is "sudo docker commit <container\_id> <username>/<imagename>"*$ sudo docker commit $CONTAINER\_ID myname/kickassapp

## **推一个镜像到它的资料库**

In order to push an image to its repository you need to have committed your container to a named image (see above)

Now you can commit this image to the repository designated by its name or tag.

*# format is "docker push <username>/<repo\_name>"*$ sudo docker push myname/kickassapp

## **可信构建**

Trusted Builds automate the building and updating of images from GitHub, directly on docker.io servers. It works by adding a commit hook to your selected repository, triggering a build and update when you push a commit.

### **设置可信构建**

1. Create a [Docker Index account](https://index.docker.io/) and login.
2. Link your GitHub account through the Link Accounts menu.
3. [Configure a Trusted build](https://index.docker.io/builds/).
4. Pick a GitHub project that has a Dockerfile that you want to build.
5. Pick the branch you want to build (the default is the master branch).
6. Give the Trusted Build a name.
7. Assign an optional Docker tag to the Build.
8. Specify where the Dockerfile is located. The default is /.

Once the Trusted Build is configured it will automatically trigger a build, and in a few minutes, if there are no errors, you will see your new trusted build on the Docker Index. It will will stay in sync with your GitHub repo until you deactivate the Trusted Build.

If you want to see the status of your Trusted Builds you can go to your [Trusted Builds page](https://index.docker.io/builds/) on the Docker index, and it will show you the status of your builds, and the build history.

Once you’ve created a Trusted Build you can deactive or delete it. You cannot however push to a Trusted Build with the docker push command. You can only manage it by committing code to your GitHub repository.

You can create multiple Trusted Builds per repository and configure them to point to specific Dockerfile‘s or Git branches.

## **私有库**

Right now (version 0.6), private repositories are only possible by hosting [your own registry](https://github.com/dotcloud/docker-registry). To push or pull to a repository on your own registry, you must prefix the tag with the address of the registry’s host, like this:

*# Tag to create a repository with the full registry location.# The location (e.g. localhost.localdomain:5000) becomes# a permanent part of the repository name*

sudo docker tag 0u812deadbeef localhost.localdomain:5000/repo\_name

*# Push the new repository to its home location on localhost*

sudo docker push localhost.localdomain:5000/repo\_name

Once a repository has your registry’s host name as part of the tag, you can push and pull it like any other repository, but it will ****not**** be searchable (or indexed at all) in the Central Index, and there will be no user name checking performed. Your registry will function completely independently from the Central Index.

参见

[Docker Blog: How to use your own registry](http://blog.docker.io/2013/07/how-to-use-your-own-registry/)

## **认证文件**

The authentication is stored in a json file, .dockercfg located in your home directory. It supports multiple registry urls.

docker login will create the “<https://index.docker.io/v1/>” key.

docker login https://my-registry.com will create the “[https://my-registry.com](https://my-registry.com/)” key.

For example:

{

**"https://index.docker.io/v1/"**: {

**"auth"**: "xXxXxXxXxXx=",

**"email"**: "email@example.com"

},

**"https://my-registry.com"**: {

**"auth"**: "XxXxXxXxXxX=",

**"email"**: "email@my-registry.com"

}}

The auth field represents base64(<username>:<password>)

# 实例

# **Hello World**

# **检测Docker安装情况**

This guide assumes you have a working installation of Docker. （这分指南假设你已经安装和运行了docker）To check your Docker install, run the following command:（你可以运行一下命令来检测docker是否安装）

*# Check that you have a working install*

docker info

If you get docker: command not found or something like /var/lib/docker/repositories: permission denied you may have an incomplete Docker installation or insufficient privileges to access docker on your machine.

Please refer to *[安装](http://docker-doc.readthedocs.io/zh_CN/latest/installation/index.html" \l "installation-list)* for installation instructions.

## **Hello World**

注解

* This example assumes you have Docker running in daemon mode. （这个例子假设你已经运行docker在daemon模式）For more information please see *[检测Docker安装情况](http://docker-doc.readthedocs.io/zh_CN/latest/examples/hello_world.html" \l "running-examples)*.
* ****If you don’t like sudo**** then see *[Giving non-root access](http://docker-doc.readthedocs.io/zh_CN/latest/installation/binaries.html" \l "dockergroup)*

This is the most basic example available for using Docker.（这个是使用docker的最基本的实例）

Download the base image which is named ubuntu: 下载基本的图像，命名为ubuntu

*# Download an ubuntu image*

sudo docker pull ubuntu

Alternatively to the ubuntu image(对ubuntu的image有多中选择）, you can select busybox, a bare minimal Linux system. （你可以busybox，一个独立的微型linux系统）The images are retrieved from the Docker repository.（这个image是从docker仓库中拉取得）

sudo docker run ubuntu /bin/echo hello world

This command will run a simple echo command, that will echo hello world back to the console over standard out.

这个命令将会运行一个简单的echo命令，将会向终端输出helloe world

****Explanation: 注解：****

* ****“sudo”**** execute the following commands as user *root*
* ****“docker run”**** run a command in a new container

Docker run命令在一个新的容器里执行一个命令

* ****“ubuntu”**** is the image we want to run the command inside of.

Ubuntu是我们想要在里面运行这个命令的image

* ****“/bin/echo”**** is the command we want to run in the container

/bash/echo 是我们希望子容器中运行的命令

* ****“hello world”**** is the input for the echo command

Hello是echo 命令的输入

****Video:****

See the example in action

## **Hello World Daemon**

注解

* This example assumes you have Docker running in daemon mode. （这条命令假设你已经安装并运行了一个docker）For more information please see *[检测Docker安装情况](http://docker-doc.readthedocs.io/zh_CN/latest/examples/hello_world.html" \l "running-examples)*.
* ****If you don’t like sudo**** then see *[Giving non-root access](http://docker-doc.readthedocs.io/zh_CN/latest/installation/binaries.html" \l "dockergroup)*

And now for the most boring daemon ever written! 现在开始有史以来最厌烦的daemon

We will use the Ubuntu image to run a simple hello world daemon that will just print hello world to standard out every second. It will continue to do this until we stop it.

（我们将会利用ubuntu的image来运行一个简单的hello world 实例，这个例子将会美妙终端输出hello world。它将不停的输出直到我们将他停止）

****Steps:****

CONTAINER\_ID=**$(**sudo docker run -d ubuntu /bin/sh -c "while true; do echo hello world; sleep 1; done"**)**

We are going to run a simple hello world daemon in a new container made from the ubuntu image.

（从ubuntu 的image 简历里一个新的容器，我们将会在这个容器里运行一个简单的hello world 实例）

* ****“sudo docker run -d “**** run a command in a new container. We pass “-d” so it runs as a daemon.

这条命你来在一个新的容器里面执行一条命令，我们传递参数-d以便于它作为一个daemon来运行

* ****“ubuntu”**** is the image we want to run the command inside of.

Ubuntu是我们想要运行名命令的容器

* ****“/bin/sh -c”**** is the command we want to run in the container

/bin/sh -c 是我们希望在容器里运行的命令。

* ****“while true; do echo hello world; sleep 1; done”**** is the mini script we want to run, that will just print hello world once a second until we stop it.

****while true; do echo hello world; sleep 1; done是我们希望运行的一个简单的脚本，这个脚本将会美妙输出hello world 直到我们停止它****

* ****$CONTAINER\_ID**** the output of the run command will return a container id, we can use in future commands to see what is going on with this process.

****$CONTAINER\_ID是命令输出的一部分，它将会返回一个容器的id，我们能够使用这一id来进一步查看这个进程的正在干什么。****

sudo docker logs $CONTAINER\_ID

Check the logs make sure it is working correctly. 检查logs以确定它工作正常

* ****“docker logs****” This will return the logs for a container 这条命令将会返回一个容器的logs
* ****$CONTAINER\_ID**** The Id of the container we want the logs for. 我们打算查看logs的容器id

sudo docker attach -sig-proxy=false $CONTAINER\_ID

Attach to the container to see the results in real-time. 附加到容器以实时查看结果

* ****“docker attach****” This will allow us to attach to a background process to see what is going on.
* ****“-sig-proxy=false”**** Do not forward signals to the container; allows us to exit the attachment using Control-C without stopping the container.
* ****$CONTAINER\_ID**** The Id of the container we want to attach too.

Exit from the container attachment by pressing Control-C.

sudo docker ps

Check the process list to make sure it is running.

* ****“docker ps”**** this shows all running process managed by docker

sudo docker stop $CONTAINER\_ID

Stop the container, since we don’t need it anymore.

* ****“docker stop”**** This stops a container
* ****$CONTAINER\_ID**** The Id of the container we want to stop.

sudo docker ps

Make sure it is really stopped.

****Video:****

See the example in action

The next example in the series is a *[Python Web应用](http://docker-doc.readthedocs.io/zh_CN/latest/examples/python_web_app.html" \l "python-web-app)* example, or you could skip to any of the other examples:

* *[Python Web应用](http://docker-doc.readthedocs.io/zh_CN/latest/examples/python_web_app.html" \l "python-web-app)*
* *[Node.js Web应用](http://docker-doc.readthedocs.io/zh_CN/latest/examples/nodejs_web_app.html" \l "nodejs-web-app)*
* *[Redis服务](http://docker-doc.readthedocs.io/zh_CN/latest/examples/running_redis_service.html" \l "running-redis-service)*
* *[SSH守护进程服务](http://docker-doc.readthedocs.io/zh_CN/latest/examples/running_ssh_service.html" \l "running-ssh-service)*
* *[CouchDB 服务](http://docker-doc.readthedocs.io/zh_CN/latest/examples/couchdb_data_volumes.html" \l "running-couchdb-service)*
* *[PostgreSQL服务器](http://docker-doc.readthedocs.io/zh_CN/latest/examples/postgresql_service.html" \l "postgresql-service)*
* *[创建MongoDB镜像](http://docker-doc.readthedocs.io/zh_CN/latest/examples/mongodb.html" \l "mongodb-image)*