

🔗 动手实战学Docker (/courses/498)

Docker 概念及基本用法

1. 课程说明

课程为纯动手实验教程，为了能说清楚实验中的一些操作会加入理论内容。理论内容我们不会写太多，已经有太多好文章了，会精选最值得读的文章推荐给你，在动手实践的同时扎实理论基础。

实验环境中可以联网，不受实验楼网络限制。

2. 学习方法

实验楼的 Docker 课程包含 14 个实验，每个实验都提供详细的步骤和截图。适用于有一定Linux系统基础，想快速上手 Docker 的同学。

学习方法是多实践，多提问。启动实验后按照实验步骤逐步操作，同时理解每一步的详细内容。

如果实验开始部分有推荐阅读的材料，请务必先阅读后再继续实验，理论知识是实践必要的基础。

3. 本节内容简介

本实验中我们初步接触Docker的概念和基本用法。需要依次完成下面几项任务：

1. Docker 基本概念
2. 安装 Docker
3. Docker 运行 Hello World


4. 推荐阅读

本节实验推荐先阅读下述内容：

- 4.1 深入浅出Docker（一）：Docker核心技术预览
(<http://www.infoq.com/cn/articles/docker-core-technology-preview>)

这篇文章介绍了Docker产生的技术发展历程，Docker中的核心技术以及相关的子项目，非常好的入门资料。

- 4.2 Understand the architecture (<https://docs.docker.com/engine/understanding-docker/>)

这篇Docker 官方的文章详细介绍了Docker的运行机制和必要的组件。不涉及到很底层的技术，可以做为对Docker的一个初步了解。
 [动手实战学Docker \(courses/498\)](https://github.com/docker/docker/blob/master/courses/498)

5. Docker 概念

5.1 容器技术

Linux 容器技术很早就有了，比较有名的是被集成到主流 Linux 内核中的 LXC 项目。容器通过对操作系统的资源访问进行限制，构建成独立的资源池，让应用运行在一个相对隔离的空间里，同时容器间也可以进行通信。

容器技术对比虚拟化技术，容器比虚拟化更轻量级，对资源的消耗小很多。容器操作也更快捷，启动和停止都要比虚拟机快。但容器需要与主机共享操作系统内核，不能像虚拟机那样运行独立的内核。

Docker 是一个基于 LXC 技术构建的容器引擎，基于 GO 语言开发，遵循 Apache2.0 协议开源。Docker 的发展得益于为使用者提供了更好的容器操作接口。包括一系列的容器，镜像，网络等管理工具，可以让用户简单的创建和使用容器。

Docker 支持将应用打包进一个可以移植的容器中，重新定义了应用开发，测试，部署上线的过程，核心理念就是 Build once, Run anywhere。

Docker 容器技术的典型应用场景是开发运维上提供持续集成和持续部署的服务。

下面我们开始介绍 Docker 中的几个基本概念。

5.2 镜像

Docker 的镜像概念类似于虚拟机里的镜像，是一个只读的模板，一个独立的文件系统，包括运行容器所需的数据，可以用来创建新的容器。

镜像可以基于 Dockerfile 构建，Dockerfile 是一个描述文件，里面包含若干条命令，每条命令都会对基础文件系统创建新的层次结构。

用户可以通过编写 Dockerfile 创建新的镜像，也可以直接从类似 github 的 Docker Hub 上下载镜像使用。

5.3 容器

Docker 容器是由 Docker 镜像创建的运行实例。Docker 容器类似虚拟机，可以支持的操作包括启动，停止，删除等。每个容器间是相互隔离的，但隔离的效果比不上虚拟机。容器中会运行特定的应用，包含特定应用的代码及所需的依赖文件。

在 Docker 容器中，每个容器之间的隔离使用 Linux 的 CGroups 和 Namespaces 技术实现的。其中 CGroups 对 CPU，内存，磁盘等资源的访问限制，Namespaces 提供了环境的隔离。

5.4 仓库

如果你使用过 git 和 github 就很容易理解 Docker 的仓库概念。Docker 仓库相当于一个 github 上的代码库。

Docker 仓库是用来包含镜像的位置，Docker 提供一个注册服务器（Registry）来保存多个仓库，每个仓库又可以包含多个具备不同 tag 的镜像。Docker 运行中使用的默认仓库是 Docker Hub 公共仓库。

仓库支持的操作类似 git，创建了新的镜像后，我们可以 push 提交到仓库，也可以从指定仓库 pull 拉取镜像到本地。

6. 安装

Docker 有两个版本，Community Edition(CE) 和 Enterprise Edition(EE)。即社区版和企业版本。我们将介绍 Ubuntu 中社区版的安装过程。

在实验环境中，已经安装了 docker-ce。

6.1 设置存储库

首先更新 apt 软件包索引：

```
$ sudo apt-get update
```

安装一些软件包，以允许 apt 通过 HTTPS 使用存储库：

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

这里我们使用阿里云提供的源，添加相应的密钥：

```
$ curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -
```

添加相应源的信息：

```
$ sudo add-apt-repository "deb [arch=amd64] http://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb_release -cs) stable"
```

6.2 安装 docker-engine

动手实战学 Docker (/courses/498)

查看此时 docker 的版本信息:

```
# 更新 apt 索引库
$ sudo apt-get update

# 查看可用的版本
$ sudo apt-cache madison docker-engine
```

更新版本的 docker 软件包为 docker-ce。我们环境中已经有 docker 了，可以先移除再进行安装。

```
$ sudo apt-get remove docker docker-engine docker.io
```

最后我们执行安装命令，我们这里是安装的 17.05 版本的：

```
$ sudo apt-get install docker-engine=17.05.0~ce-0~ubuntu-trusty
```

如果要安装最新版，可使用 `sudo apt-get install docker-ce`。

在安装成功后，Docker 的守护进程自动启动，不需要手动启动服务。


此时，我们可以查看其版本信息，使用如下命令：

```
$ docker version
```

当前实验环境中的 docker 版本信息如下图所示：

```
shiyancelou:~/ $ docker version
Client:
Version:      17.05.0-ce
API version:  1.29
Go version:   go1.7.5
Git commit:   89658be
Built:        Thu May  4 22:06:06 2017
OS/Arch:      linux/amd64

Server:
Version:      17.05.0-ce
API version:  1.29 (minimum version 1.12)
Go version:   go1.7.5
Git commit:   89658be
Built:        Thu May  4 22:06:06 2017
OS/Arch:      linux/amd64
Experimental: false
shiyancelou:~/ $
```



在上图中，该命令能够正确执行，但是如果是**自己搭建的 docker 环境**，可能会提示我们没有相应的权限连接到 Docker 守护进程进行绑定的 Unix 套接字。这是因为，默认情况下，该套接字归属于 root 用户，对于其它用户只能通过 sudo 来进行访问。

因此我们如果要想 shiyancelou 用户可以执行 docker 命令，需要创建一个名为 docker 的用户组，并将我们要执行 docker 命令的用户添加到该用户组中。该用户组会在安装后自动创建，我们只需执行添加用户到 docker 用户组的操作（在实验楼的在线实验环境中已完成该设置项）：

```
$ sudo gpasswd -a shiyancelou docker
```

添加用户到一个用户组中的方式有很多，例如我们还可以使用如下命令：

```
$ sudo usermod -aG docker shiyancelou
```

在添加成功后，我们还需要重新开始一个 shell 修改才能生效。这时可以尝试打开一个新的终端或者使用如下命令：

```
$ sudo su shiyancelou
```

6.3 启动 Docker 服务

对于 Docker 的镜像仓库来说，国内访问速度较慢，我们添加一个阿里云提供的 Docker 镜像加速器。

首先，我们需要编辑 `/etc/docker/daemon.json` 文件：

👉 动手实战学Docker (/courses/498)

```
$ sudo vi /etc/docker/daemon.json
```

然后加入如下内容：

```
{  
  "registry-mirrors": ["https://n6syp70m.mirror.aliyuncs.com"]  
}
```

修改之后，需要重启 `docker` 服务，让修改生效。使用如下命令：

```
$ sudo service docker restart
```

7. Hello world

在安装之后，我们可以通过运行一个 `hello-world` 的镜像来验证 `Docker CE` 是否被正确的安装，使用如下命令：

```
$ docker run hello-world
```


该命令会下载一个名为 `hello-world` 的镜像并运行于一个容器中。当这个容器运行时，会输出一些信息并退出：

```
shiyancelou:~/ $ docker run hello-world [13:46:50]
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:66ef312bbac49c39a89aa9bcc3cb4f3c9e7de3788c944158df3ee0176d32b75
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 client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 实验楼
shiyancelou.com
```

如上图中标注出的提示信息所示，提示我们安装正确。

8. 总结

1. Docker 基本概念
2. 安装 Docker
3. Docker 运行 Hello World

本节实验主要讲解了 docker 的相关概念以及如何安装 docker。

**本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。*

下一节：Docker 容器管理 (/courses/498/labs/1704/document)