Docker 入门教程

ruanyifeng.com/blog/2018/02/docker-tutorial.html

分享按钮
作者： 阮一峰

日期： 2018年2月 9日

2013年发布至今， Docker 一直广受瞩目，被认为可能会改变软件行业。

但是，许多人并不清楚 Docker 到底是什么，要解决什么问题，好处又在哪里？本文就来详细解释，帮助大家理解它，还带有简单易懂的实例，教你如何将它用于日常开发。

一、环境配置的难题

软件开发最大的麻烦事之一，就是环境配置。用户计算机的环境都不相同，你怎么知道自家的软件，能在那些机器跑起来？

用户必须保证两件事：操作系统的设置，各种库和组件的安装。只有它们都正确，软件才能运行。举例来说，安装一个 Python 应用，计算机必须有 Python 引擎，还必须有各种依赖，可能还要配置环境变量。

如果某些老旧的模块与当前环境不兼容，那就麻烦了。开发者常常会说："它在我的机器可以跑了"（It works on my machine），言下之意就是，其他机器很可能跑不了。

环境配置如此麻烦，换一台机器，就要重来一次，旷日费时。很多人想到，能不能从根本上解决问题，软件可以带环境安装？也就是说，安装的时候，把原始环境一模一样地复制过来。

二、虚拟机

虚拟机（virtual machine）就是带环境安装的一种解决方案。它可以在一种操作系统里面运行另一种操作系统，比如在 Windows 系统里面运行 Linux 系统。应用程序对此毫无感知，因为虚拟机看上去跟真实系统一模一样，而对于底层系统来说，虚拟机就是一个普通文件，不需要了就删掉，对其他部分毫无影响。

虽然用户可以通过虚拟机还原软件的原始环境。但是，这个方案有几个缺点。

（1）资源占用多

虚拟机会独占一部分内存和硬盘空间。它运行的时候，其他程序就不能使用这些资源了。哪怕虚拟机里面的应用程序，真正使用的内存只有 1MB，虚拟机依然需要几百 MB 的内存才能运行。

（2）冗余步骤多

虚拟机是完整的操作系统，一些系统级别的操作步骤，往往无法跳过，比如用户登录。

（3）启动慢

启动操作系统需要多久，启动虚拟机就需要多久。可能要等几分钟，应用程序才能真正运行。

三、Linux 容器

由于虚拟机存在这些缺点，Linux 发展出了另一种虚拟化技术：Linux 容器（Linux Containers，缩写为 LXC）。

Linux 容器不是模拟一个完整的操作系统，而是对进程进行隔离。或者说，在正常进程的外面套了一个保护层。对于容器里面的进程来说，它接触到的各种资源都是虚拟的，从而实现与底层系统的隔离。

由于容器是进程级别的，相比虚拟机有很多优势。

（1）启动快

容器里面的应用，直接就是底层系统的一个进程，而不是虚拟机内部的进程。所以，启动容器相当于启动本机的一个进程，而不是启动一个操作系统，速度就快很多。

（2）资源占用少

容器只占用需要的资源，不占用那些没有用到的资源；虚拟机由于是完整的操作系统，不可避免要占用所有资源。另外，多个容器可以共享资源，虚拟机都是独享资源。

（3）体积小

容器只要包含用到的组件即可，而虚拟机是整个操作系统的打包，所以容器文件比虚拟机文件要小很多。

总之，容器有点像轻量级的虚拟机，能够提供虚拟化的环境，但是成本开销小得多。

四、Docker 是什么？

Docker 属于 Linux 容器的一种封装，提供简单易用的容器使用接口。它是目前最流行的 Linux 容器解决方案。

Docker 将应用程序与该程序的依赖，打包在一个文件里面。运行这个文件，就会生成一个虚拟容器。程序在这个虚拟容器里运行，就好像在真实的物理机上运行一样。有了 Docker，就不用担心环境问题。

总体来说，Docker 的接口相当简单，用户可以方便地创建和使用容器，把自己的应用放入容器。容器还可以进行版本管理、复制、分享、修改，就像管理普通的代码一样。

五、Docker 的用途

Docker 的主要用途，目前有三大类。

（1）提供一次性的环境。比如，本地测试他人的软件、持续集成的时候提供单元测试和构建的环境。

（2）提供弹性的云服务。因为 Docker 容器可以随开随关，很适合动态扩容和缩容。

（3）组建微服务架构。通过多个容器，一台机器可以跑多个服务，因此在本机就可以模拟出微服务架构。

六、Docker 的安装

Docker 是一个开源的商业产品，有两个版本：社区版（Community Edition，缩写为 CE）和企业版（Enterprise Edition，缩写为 EE）。企业版包含了一些收费服务，个人开发者一般用不到。下面的介绍都针对社区版。

Docker CE 的安装请参考官方文档。

安装完成后，运行下面的命令，验证是否安装成功。

```
$ docker version
# 或者
$ docker info
```

Docker 需要用户具有 sudo 权限，为了避免每次命令都输入`sudo`，可以把用户加入 Docker 用户组（官方文档）。

```
$ sudo usermod -aG docker $USER
```

Docker 是服务器----客户端架构。命令行运行`docker`命令的时候，需要本机有 Docker 服务。如果这项服务没有启动，可以用下面的命令启动（官方文档）。

```
# service 命令的用法
$ sudo service docker start

# systemctl 命令的用法
$ sudo systemctl start docker
```

六、image 文件

Docker 把应用程序及其依赖，打包在 image 文件里面。只有通过这个文件，才能生成 Docker 容器。image 文件可以看作是容器的模板。Docker 根据 image 文件生成容器的实例。同一个 image 文件，可以生成多个同时运行的容器实例。

image 是二进制文件。实际开发中，一个 image 文件往往通过继承另一个 image 文件，加上一些个性化设置而生成。举例来说，你可以在 Ubuntu 的 image 基础上，往里面加入 Apache 服务器，形成你的 image。

```
# 列出本机的所有 image 文件。
$ docker image ls

# 删除 image 文件
$ docker image rm [imageName]
```

image 文件是通用的，一台机器的 image 文件拷贝到另一台机器，照样可以使用。一般来说，为了节省时间，我们应该尽量使用别人制作好的 image 文件，而不是自己制作。即使要定制，也应该基于别人的 image 文件进行加工，而不是从零开始制作。

为了方便共享，image 文件制作完成后，可以上传到网上的仓库。Docker 的官方仓库 Docker Hub 是最重要、最常用的 image 仓库。此外，出售自己制作的 image 文件也是可以的。

七、实例：hello world

下面，我们通过最简单的 image 文件"hello world"，感受一下 Docker。

需要说明的是，国内连接 Docker 的官方仓库很慢，还会断线，需要将默认仓库改成国内的镜像网站，具体的修改方法在下一篇文章的第一节。有需要的朋友，可以先看一下。

首先，运行下面的命令，将 image 文件从仓库抓取到本地。

```
$ docker image pull library/hello-world
```

上面代码中，`docker image pull` 是抓取 image 文件的命令。`library/hello-world` 是 image 文件在仓库里面的位置，其中 `library` 是 image 文件所在的组，`hello-world` 是 image 文件的名字。

由于 Docker 官方提供的 image 文件，都放在library组里面，所以它的是默认组，可以省略。因此，上面的命令可以写成下面这样。

```
$ docker image pull hello-world
```

抓取成功以后，就可以在本机看到这个 image 文件了。

```
$ docker image ls
```

现在，运行这个 image 文件。

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

`docker container run` 命令会从 image 文件，生成一个正在运行的容器实例。

注意，`docker container run` 命令具有自动抓取 image 文件的功能。如果发现本地没有指定的 image 文件，就会从仓库自动抓取。因此，前面的 `docker image pull` 命令并不是必需的步骤。

如果运行成功，你会在屏幕上读到下面的输出。

```
$ docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

... ...
```

输出这段提示以后，`hello world` 就会停止运行，容器自动终止。

有些容器不会自动终止，因为提供的是服务。比如，安装运行 Ubuntu 的 image，就可以在命令行体验 Ubuntu 系统。

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

对于那些不会自动终止的容器，必须使用docker container kill 命令手动终止。

```
$ docker container kill [containID]
```

八、容器文件

image 文件生成的容器实例，本身也是一个文件，称为容器文件。也就是说，一旦容器生成，就会同时存在两个文件： image 文件和容器文件。而且关闭容器并不会删除容器文件，只是容器停止运行而已。

```
# 列出本机正在运行的容器
$ docker container ls

# 列出本机所有容器，包括终止运行的容器
$ docker container ls --all
```

上面命令的输出结果之中，包括容器的 ID。很多地方都需要提供这个 ID，比如上一节终止容器运行的`docker container kill`命令。

终止运行的容器文件，依然会占据硬盘空间，可以使用docker container rm命令删除。

```
$ docker container rm [containerID]
```

运行上面的命令之后，再使用`docker container ls --all`命令，就会发现被删除的容器文件已经消失了。

九、Dockerfile 文件

学会使用 image 文件以后，接下来的问题就是，如何可以生成 image 文件？如果你要推广自己的软件，势必要自己制作 image 文件。

这就需要用到 Dockerfile 文件。它是一个文本文件，用来配置 image。Docker 根据 该文件生成二进制的 image 文件。

下面通过一个实例，演示如何编写 Dockerfile 文件。

十、实例：制作自己的 Docker 容器

下面我以 koa-demos 项目为例，介绍怎么写 Dockerfile 文件，实现让用户在 Docker 容器里面运行 Koa 框架。

作为准备工作，请先下载源码。

```
$ git clone https://github.com/ruanyf/koa-demos.git
$ cd koa-demos
```

10.1 编写 Dockerfile 文件

首先，在项目的根目录下，新建一个文本文件`.dockerignore`，写入下面的内容。

```
.git
node_modules
npm-debug.log
```

上面代码表示，这三个路径要排除，不要打包进入 image 文件。如果你没有路径要排除，这个文件可以不新建。

然后，在项目的根目录下，新建一个文本文件 Dockerfile，写入下面的内容。

```
FROM node:8.4
COPY . /app
WORKDIR /app
RUN npm install --registry=https://registry.npm.taobao.org
EXPOSE 3000
```

上面代码一共五行，含义如下。

- `FROM node:8.4`：该 image 文件继承官方的 node image，冒号表示标签，这里标签是 `8.4`，即8.4版本的 node。
- `COPY . /app`：将当前目录下的所有文件（除了 `.dockerignore` 排除的路径），都拷贝进入 image 文件的 `/app` 目录。
- `WORKDIR /app`：指定接下来的工作路径为 `/app`。
- `RUN npm install`：在 `/app` 目录下，运行 `npm install` 命令安装依赖。注意，安装后所有的依赖，都将打包进入 image 文件。
- `EXPOSE 3000`：将容器 3000 端口暴露出来， 允许外部连接这个端口。

10.2 创建 image 文件

有了 Dockerfile 文件以后，就可以使用 `docker image build` 命令创建 image 文件了。

```
$ docker image build -t koa-demo .
# 或者
$ docker image build -t koa-demo:0.0.1 .
```

上面代码中，`-t` 参数用来指定 image 文件的名字，后面还可以用冒号指定标签。如果不指定，默认的标签就是 `latest`。最后的那个点表示 Dockerfile 文件所在的路径，上例是当前路径，所以是一个点。

如果运行成功，就可以看到新生成的 image 文件 `koa-demo` 了。

```
$ docker image ls
```

10.3 生成容器

`docker container run` 命令会从 image 文件生成容器。

docker run 也可

```
$ docker container run -p 8000:3000 -it koa-demo /bin/bash
# 或者
$ docker container run -p 8000:3000 -it koa-demo:0.0.1 /bin/bash
```

上面命令的各个参数含义如下：

> - `-p` 参数：容器的 3000 端口映射到本机的 8000 端口。
> - `-it` 参数：容器的 Shell 映射到当前的 Shell，然后你在本机窗口输入的命令，就会传入容器。
> - `koa-demo:0.0.1` ：image 文件的名字（如果有标签，还需要提供标签，默认是 latest 标签）。
> - `/bin/bash` ：容器启动以后，内部第一个执行的命令。这里是启动 Bash，保证用户可以使用 Shell。

如果一切正常，运行上面的命令以后，就会返回一个命令行提示符。

```
root@66d80f4aaf1e:/app#
```

这表示你已经在容器里面了，返回的提示符就是容器内部的 Shell 提示符。执行下面的命令。

```
root@66d80f4aaf1e:/app# node demos/01.js
```

这时，Koa 框架已经运行起来了。打开本机的浏览器，访问 http://127.0.0.1:8000，网页显示"Not Found"，这是因为这个 demo 没有写路由。

这个例子中，Node 进程运行在 Docker 容器的虚拟环境里面，进程接触到的文件系统和网络接口都是虚拟的，与本机的文件系统和网络接口是隔离的，因此需要定义容器与物理机的端口映射（map）。

现在，在容器的命令行，按下 Ctrl + c 停止 Node 进程，然后按下 Ctrl + d （或者输入 exit）退出容器。此外，也可以用 `docker container kill` 终止容器运行。

```
# 在本机的另一个终端窗口，查出容器的 ID
$ docker container ls

# 停止指定的容器运行
$ docker container kill [containerID]
```

容器停止运行之后，并不会消失，用下面的命令删除容器文件。

```
# 查出容器的 ID
$ docker container ls --all

# 删除指定的容器文件
$ docker container rm [containerID]
```

也可以使用 `docker container run` 命令的 `--rm` 参数，在容器终止运行后自动删除容器文件。

```
$ docker container run --rm -p 8000:3000 -it koa-demo /bin/bash
```

10.4 CMD 命令

上一节的例子里面，容器启动以后，需要手动输入命令`node demos/01.js`。我们可以把这个命令写在 Dockerfile 里面，这样容器启动以后，这个命令就已经执行了，不用再手动输入了。

```
FROM node:8.4
COPY . /app
WORKDIR /app
RUN npm install --registry=https://registry.npm.taobao.org
EXPOSE 3000
CMD node demos/01.js
```

上面的 Dockerfile 里面，多了最后一行`CMD node demos/01.js`，它表示容器启动后自动执行`node demos/01.js`。

你可能会问，`RUN`命令与`CMD`命令的区别在哪里？简单说，`RUN`命令在 image 文件的构建阶段执行，执行结果都会打包进入 image 文件；`CMD`命令则是在容器启动后执行。另外，一个 Dockerfile 可以包含多个`RUN`命令，但是只能有一个`CMD`命令。

注意，指定了`CMD`命令以后，`docker container run`命令就不能附加命令了（比如前面的`/bin/bash`），否则它会覆盖`CMD`命令。现在，启动容器可以使用下面的命令。

```
$ docker container run --rm -p 8000:3000 -it koa-demo:0.0.1
```

10.5 发布 image 文件

容器运行成功后，就确认了 image 文件的有效性。这时，我们就可以考虑把 image 文件分享到网上，让其他人使用。

首先，去 hub.docker.com 或 cloud.docker.com 注册一个账户。然后，用下面的命令登录。

```
$ docker login
```

接着，为本地的 image 标注用户名和版本。

```
$ docker image tag [imageName] [username]/[repository]:[tag]
# 实例
$ docker image tag koa-demos:0.0.1 ruanyf/koa-demos:0.0.1
```

也可以不标注用户名，重新构建一下 image 文件。

```
$ docker image build -t [username]/[repository]:[tag] .
```

最后，发布 image 文件。

```
$ docker image push [username]/[repository]:[tag]
```

发布成功以后，登录 hub.docker.com，就可以看到已经发布的 image 文件。

十一、其他有用的命令

docker 的主要用法就是上面这些，此外还有几个命令，也非常有用。

（1）docker container start

docker start

docker attach （恢复到容器）

前面的 `docker container run` 命令是新建容器，每运行一次，就会新建一个容器。同样的命令运行两次，就会生成两个一模一样的容器文件。如果希望重复使用容器，就要使用 `docker container start` 命令，它用来启动已经生成、已经停止运行的容器文件。

```
$ docker container start [containerID]
```

（2）docker container stop

前面的 `docker container kill` 命令终止容器运行，相当于向容器里面的主进程发出 SIGKILL 信号。而 `docker container stop` 命令也是用来终止容器运行，相当于向容器里面的主进程发出 SIGTERM 信号，然后过一段时间再发出 SIGKILL 信号。

```
$ bash container stop [containerID]
```

这两个信号的差别是，应用程序收到 SIGTERM 信号以后，可以自行进行收尾清理工作，但也可以不理会这个信号。如果收到 SIGKILL 信号，就会强行立即终止，那些正在进行中的操作会全部丢失。

（3）docker container logs

`docker container logs` 命令用来查看 docker 容器的输出，即容器里面 Shell 的标准输出。如果 `docker run` 命令运行容器的时候，没有使用 `-it` 参数，就要用这个命令查看输出。

```
$ docker container logs [containerID]
```

（4）docker container exec

`docker container exec` 命令用于进入一个正在运行的 docker 容器。如果 `docker run` 命令运行容器的时候，没有使用 `-it` 参数，就要用这个命令进入容器。一旦进入了容器，就可以在容器的 Shell 执行命令了。

```
$ docker container exec -it [containerID] /bin/bash
```

（5）docker container cp

`docker container cp` 命令用于从正在运行的 Docker 容器里面，将文件拷贝到本机。下面是拷贝到当前目录的写法。

```
$ docker container cp [containID]:[/path/to/file] .
```

非常感谢你一直读到了这里，这个系列还有下一篇，介绍如何使用 Docker 搭建真正的网站，欢迎继续阅读。

（完）

文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（创意共享3.0许可证）
- 发表日期： 2018年2月 9日

相关文章

- **2020.08.26: rsync 用法教程**
  一、简介 rsync 是一个常用的 Linux 应用程序，用于文件同步。
- **2020.08.10: 只要一行代码，实现五种 CSS 经典布局**
  页面布局是样式开发的第一步，也是 CSS 最重要的功能之一。
- **2020.07.08: SSH 证书登录教程**
  SSH 是服务器登录工具，提供密码登录和密钥登录。
- **2020.06.21: 安卓手机系统连接电视，最好的方案是什么？**
  现在，大部分人都用手机看视频。但是手机屏幕太小，不适合高清视频。电视看视频，才有更好的体验。

广告（购买广告位）

留言（97条）

写下docker的一些感觉, 做为服务端的RD, 在本地开发后要把程序放到线上, 由于各种原因本地开发的机器可能要替换等等. 那么开发环境一但改变, 就要重新为程序安装各种服务与扩展. 这些工作也许不难, 但是做为RD 不应该做重复的事. 使用docker后, 先把环境编排好后, 以后只要一条命令就可以完成环境的部署, 一劳永逸呀.

http://www.majianwei.com/?s=docker

阮老师出品，比属精品

写的很好

docker 在微服务领域优势明显 结合k8s的生态体系 对开发和运维来说真的是省了好多事 推荐一个小工具 方便查看log 进入容器shell 以及启动、重启、偏上容器的扩展命令 https://github.com/iuv/xx

我一直以为docker指的是tor里面经常使用的搜索引擎。 [狗头]

太感谢了，真是我想看什么，阮老师就写什么。:)

你好
阮一峰
如果你有兴致的话、我希望你有时间能写一篇专门针对linux容器的文章。
谢谢：）

阮老师写的很棒，我一步步操作下来了，就是这条命令需要改一下

$ docker container ls

需要改成

$ docker container ls -l

感谢分享，老师总能把一个复杂的东西讲得直白，容易理解。

一直很排斥去学习docker。现在有了阮老师的入门教程，一下子就有了学它的兴趣了。谢谢阮老师。

对于前端来说有啥用呢？

感谢阮老师！

博主你有很多文章的链接怎么都打不开啊，例如那个koa教程

作为前端需要运行复杂的后端环境的人员来说，这个简直救命了~~

> 引用llfylwg的发言：
>
> 对于前端来说有啥用呢？

为什么要对前端有用？

感觉一般

等续集!!!

写得很好，特别是命令加上了container/image

> 引用timothy的发言：
>
> 为什么要对前端有用？

不用那么狭隘，知识的深度和广度都是程序员极力追求的

等阮老师的续集...

简单清晰，给一个赞。

精品

您好，阮老师，我在执行docker image push命令的时候一直Retrying，过一段时间后提示Service Unvailable，是被墙了吗，有相应的解决方法吗

79832b4d1f2d: Retrying in 3 seconds
37b22fa5fbec: Retrying in 1 second
c5b2ac536264: Retrying in 2 seconds

> 引用iStar的发言：
>
> 太感谢了，真是我想看什么，阮老师就写什么。:)

我也是，刚刚正打开docker官网，然后打开博客就发现刚好有写

已经习惯了看阮老师总结的文章了，自己总结了一半的docker突然就没动力写下去了,我这样还有救么。。

docker container ls 是不是用 docker ps 以及 docker ps -a 更方便看 container 的运行状态呢

我们欠峰哥一个微信红包，实属小恩小惠，但是希望能够以此方式推进峰哥多出精品，以解我等菜鸟之惑。

阮老师讲东西一直都是简单易懂

Npm,docker，都感觉不是将业务简化。

太简洁明了了，大爱！

为什么我的docker找不到image、container 命令呢？

请问 docker container run 和 docker run 命令有什么区别？

老师好。
您在写到 docker container stop 这里时候，误将 docker 写成了 bash

你好，请教一个问题。我现在的情况是，有一个linux下的程序，不支持centos7，所以装了docker拉取了centos6的镜像。
现在我的程序是要安装在centos6里才能运行，我已经把程序copy到了镜像里。由于我的程序安装的时候会去读网卡设置，docker里的centos并没有ifcfg-eth0文件，所以一直进行不下去。请问有其他办法吗？

> 引用hy05190134的发言：
>
> docker container ls 是不是用 docker ps 以及 docker ps -a 更方便看 container 的运行状态呢

阮一峰老师，我觉的查看容器还是docker ps比较合适，官方推荐的也是这个

# 列出本机的所有 image 文件。
$ docker image ls

这个不对吧
docker images

写得很不错，不过就我一个用windows吗？

> 引用mufeng的发言：
>
> 阮一峰老师，我觉的查看容器还是docker ps比较合适，官方推荐的也是这个

> 引用hy05190134的发言：
>
> docker container ls 是不是用 docker ps 以及 docker ps -a 更方便看 container 的运行状态呢

docker container options 是Docker 1.13中的更新，docker container ls 与 docker ps 功能相同，但是语义更明确，简化了Docker的用法，所以更推荐使用新的写法

非常适合入门的一片文章了，喜欢阮一峰老师文章。

5和6的小标题序号是不是错了？漏了一个

$ sudo usermod -aG docker $USER

执行完之后貌似需要先logout一下，才会生效。

在实际操作的过程遇到了一些问题：

Context： 我在Window 10 enterprise 上安装的最新版本的docker

问题：在下载Hello-world image之后一直提示我(No hypervisor is present on this system)

解决方法：正确在windows中安装HyperVisor 和 Container feature之后，需要在BIOS中Enbale Virtualization. 进入BIOS中找到System Security，然后进入找到Virtualization并Enable，此操作需要重启。

之后发现官方的node image 暂时还不支持在windows上运行，大家还是使用linux吧。

写的太好了，通俗易懂！

对OS的依赖是否存在？ 就是说生成image时候的OS是否必须和以后运行容器时的物理机器的OS类型和版本号都一致？

bash container stop [containerID]这里是不是应该写成docker container stop [containerID]?

> 引用走在成功的大道上的发言：
>
> 为什么我的docker找不到image、container 命令呢？

centos 6.8 无法识别 $ docker image 命令

docker 的官方文档为什么一直打不开，有人知道怎么打开吗

老师的有一句命令写错了，如果加了CMD，不需要shell映射了，那么为啥还要写-it。
应该改成：docker container run -p 8000:3000 koa-demo

for Linux Mint 18.3 :

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable"

文中有一个错误。

$ docker image build -t koa-demo .

最后的那个点表示 Dockerfile 文件所在的路径，上例是当前路径，所以是一个点。

并不是的。.实际上是在指定上下文的目录, docker build 命令会将该目录下的内容打包交给 Docker 引擎以帮助构建镜像。

细节可查看https://yeasy.gitbooks.io/docker_practice/content/image/build.html

深入浅出，很好理解，感谢阮老师

docker底层的2个核心技术分别是Namespaces和Control groups。自1.20版本开始docker已经抛开lxc，不过对于理解docker还是有很大帮助。

看过最容易懂的一篇文章，收藏啦。

写的真不错，一下搞懂了Docker是什么，以前听人讲过不要把Docker理解成虚拟机，但是最后也没说明白它与虚拟机的差别，阮老师的说的很清晰，一目了然

如果用自定义文件夹模式，把需要的类库，依赖都打包进去，作为一个文件夹整体为容器也可以呀，，我们很多项目就是这么部署，迁移机器直接文件夹copy。。貌似找不到必须用docker的理由？？优点相比普通的文件夹容器模式到底在哪里呢？？docker相对来说麻烦，不跨win linux平台。。文件夹是跨越win linx的。。

阮老师，想咨询下，docker可以无限嵌套么？
之前看了一个docker-in-docker，这样嵌套下去会有哪些问题？

通俗易懂,赞一个

为啥我那个koa的例子还是Not found啊，好像也没有说解决方法啊，小白求教

阮老师的很多文件，都是写得通俗易懂，不错，加上我对docker的了解。谢谢。

有没有新的入门资料、cheatsheet推荐？

希望有机会看到k8s入门文章

菜鸟入门，感觉很容易明白

不吹不黑，只是发表下个人看法，像这种文章，去看官网的教程不就得了，又详细又提高英语阅读水平，一个个在低下鼓掌叫好。

> 引用hy05190134的发言：
>
> docker container ls 是不是用 docker ps 以及 docker ps -a 更方便看 container 的运行状态呢

docker container stats

$ docker image tag koa-demos:0.0.1 ruanyf/koa-demos:0.0.1
这个命令里koa-demos应改成koa-demo,多了个s

> 引用王昊天的发言：
>
> 您好，阮老师，我在执行docker image push命令的时候一直Retrying，过一段时间后提示Service Unvailable，是被墙了吗，有相应的解决方法吗
>
> 79832b4d1f2d: Retrying in 3 seconds
> 37b22fa5fbec: Retrying in 1 second
> c5b2ac536264: Retrying in 2 seconds

不知道是不是被墙了,我是开着梯子的,一切正常.

真的感谢，通俗易懂

老师，这里我还是不太清楚，如果我的服务器是没有网络环境的，那怎么从其它的机器上拷贝过来docker的img，并粘贴到服务器上进行部署使用呢？

阮老师太棒了，10min入门docker

> 引用Docker的发言：
>
> 不吹不黑，只是发表下个人看法，像这种文章，去看官网的教程不就得了，又详细又提高英语阅读水平，一个个在低下鼓掌叫好。

我觉得老师讲的很好啊，先引入问题，比如环境配置的痛点，所以有了docker这种东西。循序渐进挺好的。至于后续那些命令，确实可以去官网看文档。看博客的时候，汲取有用的东西就好！哈哈

阮老师，我看您很多文章后面都有
“版权声明：自由转载-非商用-非衍生-保持署名（创意共享3.0许可证）”
这样的版权声明，但是我不太清楚把文章转载至自己的个人公众号是否为商用，希望老师能够得到老师的答复。

> 引用xionghongzhi的发言：
>
> $ sudo usermod -aG docker $USER
>
> 执行完之后貌似需要先logout一下，才会生效。

是的,需要 log out 然后 log in 才会生效

> 引用kid1412的发言：
>
> 不知道是不是被墙了,我是开着梯子的,一切正常.

可以打开啊 不用翻墙

我才知道还有'docker container ls'和'docker image ls'这2个命令...

请教一个问题：
文章中
“这个例子中，Node 进程运行在 Docker 容器的虚拟环境里面，进程接触到的文件系统和网络接口都是虚拟的，与本机的文件系统和网络接口是隔离的，因此需要定义容器与物理机的端口映射（map）。”

端口映射 不是在在下面命令行（-p）中完成了吗？

“docker container run -p 8000:3000 -it koa-demo:0.0.1 /bin/bash”

很厉害，还有JS教程，网络协议等等，受益匪浅。

# service 命令的用法
$ sudo service docker start

# systemctl 命令的用法
$ sudo systemctl start docker

好像在macos上不可用，您知道怎么解决吗？
谢谢

我是初学者，请教一下，我想修改别人image，应该怎么操作 :)

> 引用season的发言：
>
> 我是初学者，请教一下，我想修改别人image，应该怎么操作 :)

像文章中第十节讲的那样，使用Dockerfile 通过继承已有镜像来创建新的镜像
如果怕使用Dockerfile 操作有问题的话可以在已有的镜像上做完操作后使用docker commit从容器创建镜像

十一、其他有用的命令
第（2）小节里的命令应该是
“docker container stop [containerID]”
现在写的是
“bash container stop [containerID]”

写的很好，学习到了。谢谢！

JVM屏蔽了与具体平台相关的信息
参考wikipedia：
https://zh.wikipedia.org/wiki/Java%E8%99%9A%E6%8B%9F%E6%9C%BA

从这方面说，jvm也是一种虚拟化方案了吧

> 引用子孓的发言：
>
> 不用那么狭隘，知识的深度和广度都是程序员极力追求的

docker本来就是容器化的，能放任意的程序，为什么前端项目不可以

阮老师把"镜像比作模板,容器比作实例",真的讲的很明白.
自己学习的时候看的是, 细节讲的很多, 但是概念却讲的不明白.
"镜像继承镜像"这句也是,很棒.

> 引用王昊天的发言：
>
> 您好，阮老师，我在执行docker image push命令的时候一直Retrying，过一段时间后提示Service Unvailable，是被墙了吗，有相应的解决方法吗
>
> 79832b4d1f2d: Retrying in 3 seconds
> 37b22fa5fbec: Retrying in 1 second
> c5b2ac536264: Retrying in 2 seconds

> 引用王昊天的发言：
>
> 您好，阮老师，我在执行docker image push命令的时候一直Retrying，过一段时间后提示Service Unvailable，是被墙了吗，有相应的解决方法吗
>
> 79832b4d1f2d: Retrying in 3 seconds
> 37b22fa5fbec: Retrying in 1 second
> c5b2ac536264: Retrying in 2 seconds

> 引用Just的发言：
>
> 应该是你没登录

我有个疑问，上面提到连运行的结果都保存的到image文件里面。比如，我环境里面有数据库，redis,那么我运行容器后产生的测试数据也还在么？我持续迭代项目，是要每次都要重新build一下么？

第十一节中第二部分 docker container stop 中有笔误，应该是`docker container stop [containerID]`，而不是`bash container stop [containerID]`

感谢阮老师的文章，很详细，如果可以在页面加上目录链接更好哈

执行 '10.4 > docker container run --rm -p 8000:3000 -it koa-demo:0.0.1' 会进入node终端'>_'，可能需要重新build一个新的image（docker image build -t koa-demo:0.0.2），

然后运行 docker container run --rm -p 8000:3000 -it koa-demo:0.0.2。（Docker version 19.03.5, build 633a0ea）

讲的很全面很透，又学到新知识了，感谢分享

docker image pull/push/build 与 docker pull/push/build 加image与不加image有什么区别？我不加image也可以

上直播吧，我要打赏，妈的，现在连直播吃饭都能日入上万，阮大大呕心呖血，写文章，免费公开，还有人瞎BB，啥世道。

写的贼好，看官方文档看的我一头雾水，官方文档都是默认你已经有了很多基础知识的，就是看评论一堆人各种挑剔嫌弃，真是不能理解，如今像阮老师这种好博客还有多少？？

> 引用anne的发言：
>
> 阮老师写的很棒，我一步步操作下来了，就是这条命令需要改一下
>
> $ docker container ls
>
> 需要改成
>
> $ docker container ls -l

docker container ls 可以查看运行中的docker容器

docker container ls -l : 表示查看最近创建的容器

我容器重啓了，但是怎麼才能把之前對鏡像的配置找回來？

我要发表看法

«-必填

«-必填，不公开

«-我信任你，不会填写广告链接